REAL-TIME BEARING ESTIMATION IN A MULTI-SOURCE ENVIRONMENT USING MULTI-PROCESSOR, MULTI-ALGORITHMIC ACCELERATION

BY

ANTONE LEE KUSMANOFF

Bachelor of Arts
Southern Illinois University, 1967

Bachelor of Science in Electrical Engineering
University of Missouri, 1972

Master of Science in Electrical Engineering
University of Missouri, 1973

Master of Science in Electrical Engineering
Georgia Institute of Technology, 1982

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the Degree of
DOCTOR OF PHILOSOPHY
May, 1989

REAL-TIME BEARING ESTIMATION IN A
MULTI-SOURCE ENVIRONMENT
USING MULTI-PROCESSOR,
MULTI-ALGORITHMIC
ACCELERATION

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

PREFACE

This thesis considers the application of a multi-processor computer system, using cooperating multi-algorithms, with an improved "*noise-signal subspace*" algorithm for direction-of-arrival estimation in a multi-source environment. The main driving factors are that the eigenstructure analysis computational burden and the subspace searches for linear combinations of steering vectors in typical eigenanalysis algorithms are impractical in real-time situations for large antenna arrays on a serial computer.

Applying a multi-instruction stream, multi-data stream organized computer with P processors is shown to reduce the processing time of large antenna array systems to approach an order of P less than that on the serial processor.

Greatly reducing the computation time, the typical N-cubed eigenanalysis has been replaced with an iterative N-squared procedure that has a more favorable structure for parallelization and eliminates the requirement for the complete eigensystem identification. In certain operations, using an iterative acceleration technique of cooperation between multi-algorithms within multi-processors, the time required may be yet significantly lower.

Finally, the development of a new peaking functional has allowed both the maximum signal and the minimum noise eigenvector components to present integrated information related to the direction-of-arrival. The new functional improves the multiple signal arrival resolution capabilities and requires less computational effort.

The result is a multi-processor, multi-algorithmically accelerated, super

resolution, passive array, real-time direction-of-arrival capability for large antenna systems.

In way of acknowledgement, I wish to thank the Office of Naval Research and University Center for Energy Research for the generous support that they provided to me and my family during this research. Without their support I could never have acomplished this work.

I wish to thank my committee members for their readings and suggestions of improvement at each stage of this work.

I wish to thank Dr. Baker who allowed me to begin this program and whose support, advice, and confidence has allowed me to complete this work.

I wish to especially thank Dr. Yarlagadda, my advisor and mentor, whose infinite wisdom, unlimited patience, and clever guidance were critical components to the success that I have had with this dissertation.

I can never thank enough my parents, Boris and Evelyn Kusmanoff, for their life long understanding and for always being there when I needed them.

I wish to thank my children, Stacey, Jason and Mary for unselfishly giving me the time necessary to do this research and thesis.

I thank Dianne, my devoted wife, to whom I am her devoted husband, for her support, belief, encouragement, and unending love, who agreed to share this difficult journey knowing the trials and difficulties ahead. Without you, this work, as my life, would be without meaning.

Praise the Lord, and thank God, it is finished because it is his will.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\underline{f}$	- column vector f
$[a, b, \ldots, c]^T$	- column vector with entries a, b, . . . ,c
f_c	- carrier frequency
$\mathbf{A}$	- matrix **A**
$\mathbf{A}^T$	- transpose of **A**
$\mathbf{A}^H$	- complex conjugate transpose of **A**
$\lvert\mathbf{A}\rvert$	- determinant of **A**
$\underline{v} \cdot \underline{z}$	- inner product of vectors $\underline{v}$ and $\underline{z}$
$E\{\ \}$	- expectation operator
θ_m	- the mth source's direction-of-arrival
M	- number of arriving wavefronts
N	- number of antennas in array
S	- number of samples or speedup
D	- number of investigated angles
I	- number of iterations
λ_c	- wavelength of carrier
λ_i	- Eigenvalues, i = 1, 2, . . . , N
λ_{max}	- Maximum eigenvalue
λ_{min}	- Minimum eigenvalue
P	- number of processors in a parallel computer
S_ρ	- speedup ratio
S_δ	- speedup differential

CHAPTER I

INTRODUCTION

The problem of locating the correct bearings of incoming signals with a passive antenna array has been the topic for many years by many authors. Physical applications are found in the diverse areas of sonar, seismology, radar, and radio-astronomy to name a few. As is well known, direction finding is analogous to frequency spectrum estimation except that the signals are functions of samples in a space aperture rather than in a time aperture. The sought after quantity in direction finding is the *wavenumber* which can be converted to a signal bearing knowing the antenna geometry (Marple, 1987).

The application of eigenvector decomposition to this array signal processing problem is a more recent advancement and it has been shown to enhance the direction of arrival resolution capabilities compared to classical beamforming methods (Schmidt, 1981, Speiser, 1987). The resolution improvement is attained at the expense of a rather large computational burden (Johnson, 1982). These algorithms use the special eigenstructure of the sample signal covariance matrix computed from the output of the array receivers. Many algorithms have been developed for estimating the *direction-of-arrival* (DOA) of multiple wavefronts using signal and noise subspace methods including Schmidt (1981), Owsley (1981), and Johnson and Degraaf (1982).

A combination of a need for greater stand-off range for the passive receivers, the existence of more ambient noise sources, and quieter operation of the

targets, has led passive systems toward longer arrays or multiple arrays among other improvements (Marple, 1987). However, the computational burden referred to above increases with the cube of the number of antenna elements. Therefore as the number of antenna elements increase, the eigenvector super resolution techniques quickly become batch mode operations, no longer applicable to the on-line or real-time situations.

Although a review of the DOA problem and a detailed look at one widely used procedure will be presented, the aim of this research is not to provide an exhaustive survey of the previous work, nor is it to introduce a radically new DOA estimator. Rather, it is to present a significant advancement arrived at by the integration of several eigenanalysis DOA methods and parallel processing.

The first component of the advancement is in the form of a high speed serial algorithm that greatly lowers the heavy computational eigenstructure analysis burden as well as the other computations involved. Then, further decreasing the computer time used, a combination of parallelization, and an enhanced parallelization technique is shown to obtain the objective of super resolution, multi-signal, direction finding in real-time using large passive antenna arrays.

The concept of developing a fast algorithm is usually thought of as providing a computational procedure that resolves the solution in a shorter time by reducing the number of operations. This can also be accomplished by applying several concurrently operating processors to the problem. Most often these efforts result in replacing that which is conceptually clear and traditional with what is computationally efficient (Blahut, 1985). This research shows that an additional speedup gain can be reached when a set of different algorithms are combined into a single concurrent system that results in a new parallel procedure. The new parallel procedure can be faster than the fastest serial algorithm out of the set converted to a parallel algorithm. This is possible when

different data situations cause different and unpredictable responses within the algorithms. This is accomplished through a synergistic-like action between the different algorithms being employed simultaneously and communicating the solution progress between algorithms.

Although improved speed to reach on-line performance is the primary driving factor, the gain in speed by substituting algorithms and parallelization is meant to be accomplished without the significant loss of accuracy compared to similar methods. *Bias* and *variance* are always basic tradeoffs in spectral estimation and are likewise considerations in DOA problems.

The genesis for the new serial and parallel procedures developed here lies with a DOA algorithm called *MUltiple SIgnal Characterization* or MUSIC (Schmidt, 1981). A. O. Schmidt discovered that the elements of the minimum eigenvector of the array cross spectral matrix were shown to be interpretable as coefficients of a polynomial whose roots provide the source directions (Schmidt, 1981). His work and Pisarenko's *harmonic retrieval* method for the time series version of the same problem lays the groundwork for this research (Pisarenko, 1973). Similar work using the maximum eigenvectors was a critical inclusion to this research's ultimate algorithm (Reddi, 1979, Cadzow, 1988).

The resulting new algorithm has been finalized into a multi-processor, multi-algorithmic procedure. This algorithm provides an improved solution, in near real-time for even very large antenna arrays.

Direction Finding Problem

The basic problem of determining directions-of-arrival of multiple incoming signals from a set of noisy observations requires the solution to a set of overdetermined system of equations that can be represented as a vector-matrix

relationship shown as Equation (1-1).

$$\underline{x} = \mathbf{A}\,\underline{f} + \underline{w} \tag{1-1}$$

Here $\underline{x}$ is a column vector (notation will have vectors underlined, using lower case letters, and matrices boldfaced, using the upper case) that represents the collection of output vectors of observed data that is generated by the intersection of the receive antenna array geometry (referred to as the *array manifold*) and the bearings of the incoming plane waves. The objective is to solve the set of linear equations to determine the M different DOA bearings. Naturally, imperfect knowledge of the array manifold will affect the follow on estimation. There are other algorithmic procedures being developed to correct problems associated with sensor positioning errors, however the calibration of the array will be assumed as a result of accurate measurement and storage of the data to analytically overcome this problem (Seymour, 1987).

At any instant, the sth *snapshot* vector (out of S samples per antenna) would be defined as:

$$\underline{x}_S=[x_S(1)x_S(2)\bullet\bullet\bullet x_S(N)]^T\,, \tag{1-2}$$

where N is the number of antennas, and "T" denotes transposition. The vector, $\underline{f}$, is the noise free input *signal-in-space* vector composed of the multiple incoming signals. The elements of the input signal component, $\underline{f}$, are generated at the sources. The entries of the matrix **A** describe the relationship of the phase shifts due to the geometry between the antenna system and the various directions-of-arrival of the received signals. Included is another column vector, $\underline{w}$, which represents the total additive noise. The noise is assumed to be uncorrelated with the incoming signals and uncorrelated between antennas.

As a preamble to direction finding, Equation (1-1) will be established using a colinear equally spaced antenna array consisting of N omnidirectional receiver elements with M (M<N) plane waves arriving from M distinct directions. Each of

the M plane waves is assumed to have the same carrier frequency, f_c, and to be defined *narrowband* and incoherent. It is possible to consider the broadband frequency problem as several non-overlapping narrowband problems, and then apply narrowband subspace processing to each band (Wax, 1985). There are newer broadband approaches currently being developed based on eigenstructure procedures that show promise of advancements related to this problem (Buckley, 1988, Shaw, 1987). Limiting the scope of this study, this research will only consider the narrowband problem.

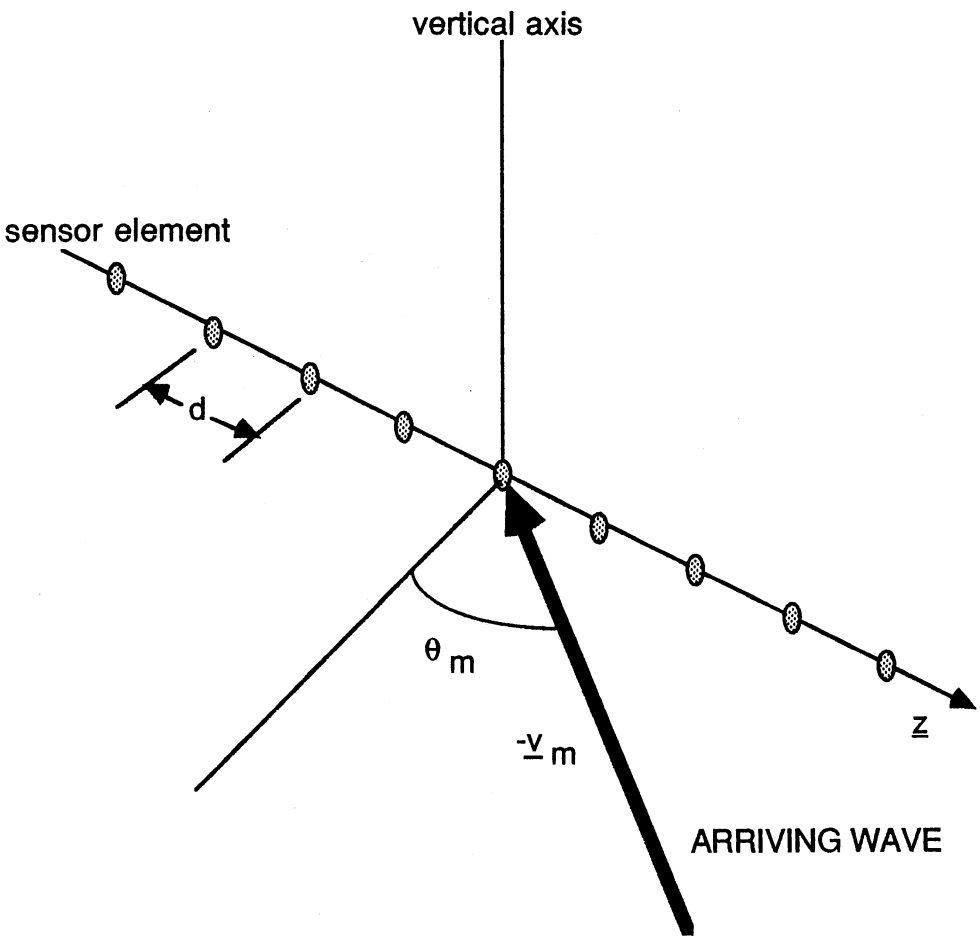
Shown in Figure 1 is the mth signal (of the M signals), assumed to be coming into the array at a direction defined by vector $-\underline{v}_m$ which is the vector wavenumber for that mth plane wave. The figure shows an element at the center of the array, but this is not a physical requirement for an actual system. When N is even, there will not be a real sensor element at this location, however this will not be important to the final equations. The received signal value of the mth wave differing by the phase shift between antenna elements can be expressed as the output at the nth (out of N total) element adjusted to the center of the array in phase and amplitude as follows (Haykin , 1985):

$$s(n,m,t)= A_m \cos\left[2\pi f_c t + 2\pi(n-(N+1)/2)\underline{v}_m \bullet \underline{z} + \alpha_m\right] \qquad (1\text{-}3)$$

where:

n = 1,2,...,N the number of elements in the array
m = 1,2,...,M the number of arriving plane waves
f_c = the carrier frequency of the plane waves
t = time
$\underline{v}_m$ = the vector wave number of the mth arriving wave
$\underline{z}$ = unit vector along the line of the array
A_m = Amplitude of the signal s(n,m,t)
α_m = Phase of the signal s(n,m,t) measured at the center of the array

The dot product of a unit vector along the antenna array axis and the vector defining the arrival direction of the plane wave, represented above by $\underline{v}_m \bullet \underline{z}$, can be expressed in terms of the distance between the array elements, d, the



θ_m = azimuth angle of arrival of the mth plane wave

d = spacing between elements

$\underline{z}$ = unit vector along the line of the array

$-\underline{v}_m$ = vector wave number of the mth arriving wave

Figure 1. Determination of Azimuth Angle Using a Linear Array

arrival angle θ_m, and wavelength, λ, of the arriving wave. The incoming wave is assumed to be propagating at a constant speed, C, here assumed to be the speed of light, hence the dot product is represented as:.

$$\underline{v}_m \cdot \underline{z} = (d/\lambda) \sin \theta_m \qquad (1\text{-}4)$$

Next, the *electrical phase angle*, Φ, between elements along the array is defined as a function of the incoming angle, θ_m , by:

$$\Phi_m = (2\pi d/\lambda) \sin \theta_m . \qquad (1\text{-}5)$$

Using a colinear array as Figure 1 depicts, allows a direction-of-arrival ambiguity to exist. Any arriving wave making an angle θ_m with the axis of the array coinciding with the same cone angle around the axis, will have an identical electrical phase shift, Φ_m. The ambiguity does not exist if the array is planer and a three dimensional pointing vector as a function of azimuth and elevation is substituted in the mathematical development above.

It is also possible to separate the arriving angles into unique azimuth and elevation angles if the antenna configuration is a simple *crossed array*. This requires solving for each antenna branch's angle-of-arrival and use of the geometry involved to establish a pointing vector toward the direction-of-arrival. Figure 2 illustrates this procedure with a crossed array (Kaplan , 1987). The azimuth and the elevation are determined by estimating the angle-of-arrival for each array arm, and then the three dimensional pointing vector DOA is computed from the intersection of the cone angles.

The straight three dimensional approach causes the largest number of computations because it requires an azimuth sweep for every elevation angle investigated. The intersecting cone angle approach is much faster, but an ambiguity cab occur when multiple wavefront arrivals exist.

Being careful to point out any loss of generality, the simplified geometry of Figure 1, rather than Figure 2, or a planar array, will be used for development of

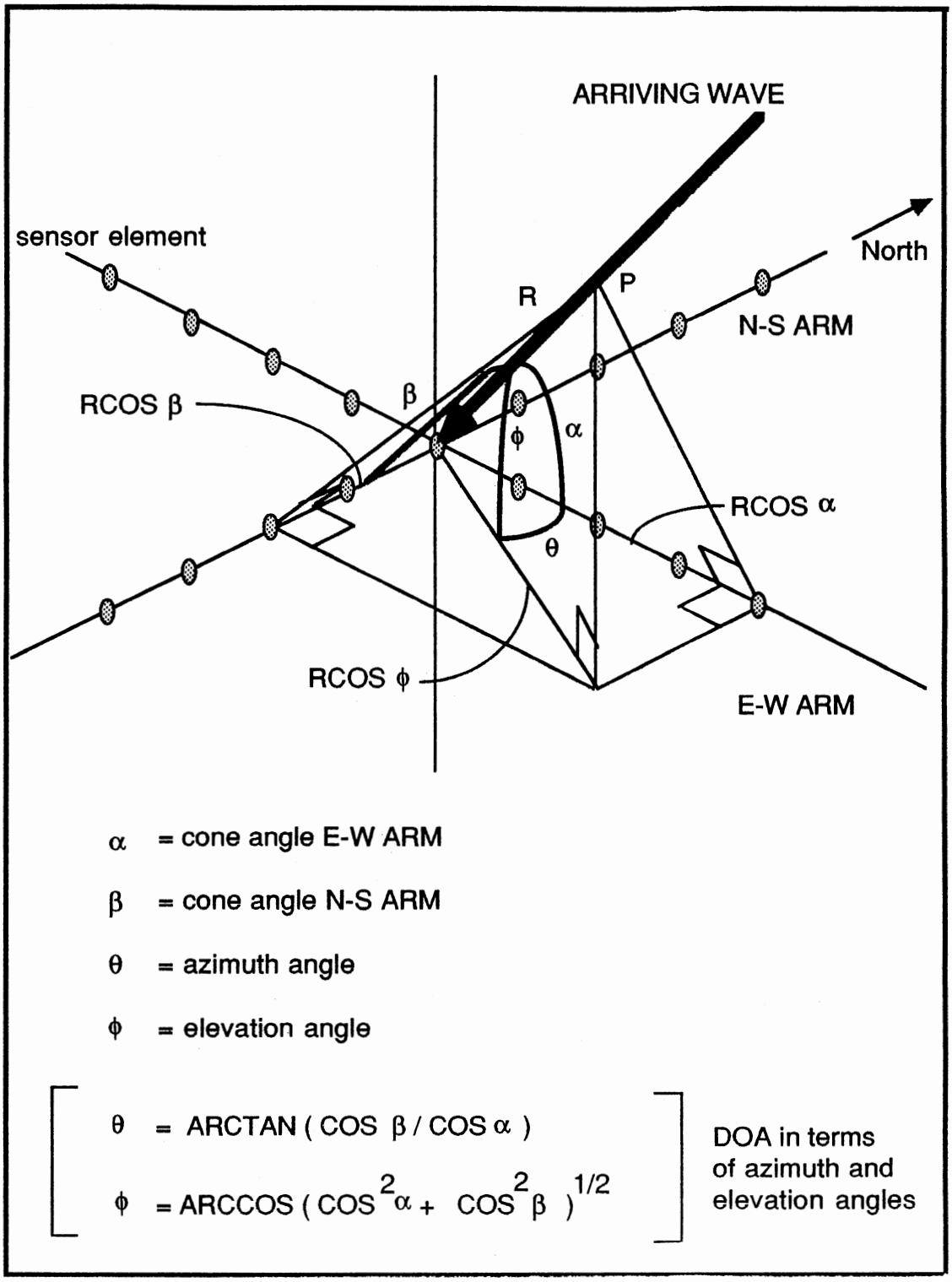


Figure 2. Determination of Azimuth and Elevation Using a Crossed Array

the DOA capability. It should be noted, however, that in Chapter IX the above cone angle procedures outlined for Figure 2 are used to determine both the azimuth and elevation for the DOA in a crossed array example.

Continuing the colinear signal model formulation, let a_m be the complex amplitude of the signal, s(n,m,t), measured at the center of the colinear array where n equals (N+1)/2. As stated earlier, if N is an odd number of elements, then this will not be a real sensor location. If N is an even number of elements, the center of the array will not be at an actual sensor location, but this will be seen to not be important to the final equations. In any event, using phasor representation yields a simplified form of the signal variable independent of the time, as:

$$s(n,m) = f_m \exp [\, j\,(n-(N+1)/2\,)\,\Phi_m\,] \qquad (1\text{-}6)$$

where, $f_m=A_m \exp(\, j\,\alpha_m)$.

The narrowband noise, a narrowband filtered version of the white noise at the input to the antennas, can be represented in a similar manner where the amplitude, B, would be *Rayleigh* distributed and the phase, β, would be *uniformly* distributed over the range (0, 2π).

$$w(n,t) = B_n \cos(2\pi f_c t+\beta_n) \qquad (1\text{-}7)$$

The noise can also be represented independent of time with the phasor simplification to yield ,

$$w(n)=B_n \exp(j\beta_n)\,, \qquad (1\text{-}8)$$

a complex valued random variable that typically has a zero mean and is *Gaussian* distributed.

To expand the problem to include the multiple signal arrivals, Equation (1-6) is summed over the set of M plane waves (along with the added noise) to describe the observed signal at one typical antenna element, n. Figure 3 illustrates the final expression of Equation (1-1) in an expanded matrix form to

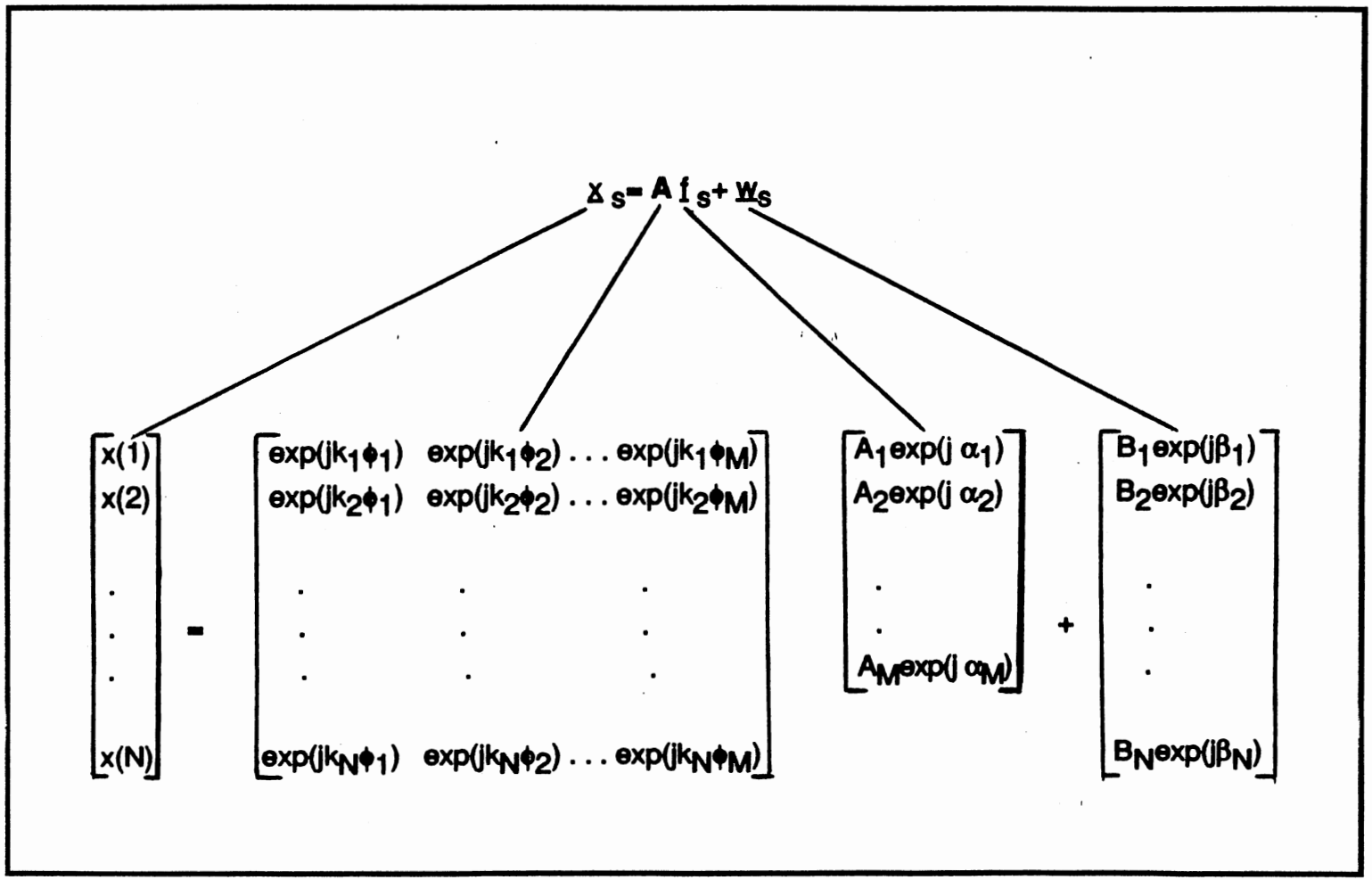


Figure 3. Expanded Matrix Form of Equation (1-1)

represent the N antenna element outputs for the set of M arriving plane waves. The symbol $k_n = n-(N+1)/2$ is used as a function of antenna position to simplify the expression further. The s subscript in Figure 3 is a reminder that the vector $\underline{x}$ is actually a set of S observations of independent measurements which will allow temporal averaging to improve the signal to noise ratio (SNR).

During the processing interval, the matrix **A** is assumed to be stationary (which accounts for **A** not having the s subscript), which in turn requires that the directions-of-arrival to not change significantly during the taking of the S snapshots (which affects the length of time for sampling). The noise and the signal-in-space vector are expected to vary unpredictably over the sample period and are considered stochastic processes because of their behavior.

The target of DOA eigenanalysis is the spatial correlation matrix also called the covariance matrix of the observed vector $\underline{x}$ which is defined as,

$$\mathbf{R} = E\{\underline{x}\,\underline{x}^H\} \quad , \tag{1-9}$$

where the superscript "H" denotes complex conjugate transpose, and $E\{\bullet\}$ represents the expectation operation (Haykin, 1985). Of course, only a finite number of snapshot samples are ever available, so an estimate of **R** which will be called the *sample covariance matrix* is computed, as the average of the outer products of the S different $\underline{x}$ snapshot vectors.

$$\mathbf{R}_X = \frac{1}{S} \sum_{S} [\underline{x}_S\,\underline{x}_S^H] \tag{1-10}$$

Thus, given the S samples from which the estimate $\mathbf{R}_X$ can be computed, the problem is to determine each of the directions-of-arrival, θ_m, of the M waves. These directions will be seen to be related to the eigenstructure of $\mathbf{R}_X$, the eigenvalues and their associated eigenvectors of $\mathbf{R}_X$.

Achieving high resolution for this problem has been addressed with many techniques, but the signal subspace and noise subspace procedures have

been found to be the best unbiased asymtotically error free approaches (Reddi, 1979, Schmidt, 1981, Johnson, 1982, Cadzow, 1988, Kumaresan, 1988). However, simultaneously with the improvement in resolution a new kind of burden arose. This burden is the large number of computations required to resolve the waves (Schmidt, 1981). Parallelization of the algorithm on a modern parallel processor is one way suggested to speedup the solution (Reddi, 1979, Speiser, 1985). This step was an objective in the initial goal of this research. However, it was concluded that because of the multiple N-cubed order of computations that are required, additional serial algorithmic advancements would be necessary to reach real-time speeds except for small antenna arrays. The particular serial advancements obtained will be unfolded in later chapters, however, and the next introductory topic will be to introduce the basic capabilities of parallel processing.

General Parallel Processing

Recently with the development of microcomputer based parallel architecture such an Intel Scientific Computers' IPSC/2 , it has become possible to place what is equivalent to supercomputing power in locations normally limited to minicomputer capacities (Intel Corporation, 1986). The IPSC/2 configuration has multiple processors in the hardware configuration of a *hypercube* where each processor is directly connected to $\log_2 P$ other processors through an internodal communications network. Here and throughout, P represents the number of processors in the parallel system. Figure 4 shows the nodal connectivity of concurrent hardware nodal configuration of a hypercube of 32 nodes. This organization is defined as being a cube of *dimension* 5 (where 2^5=32 nodes). Computers of this kind and other parallel organizations can

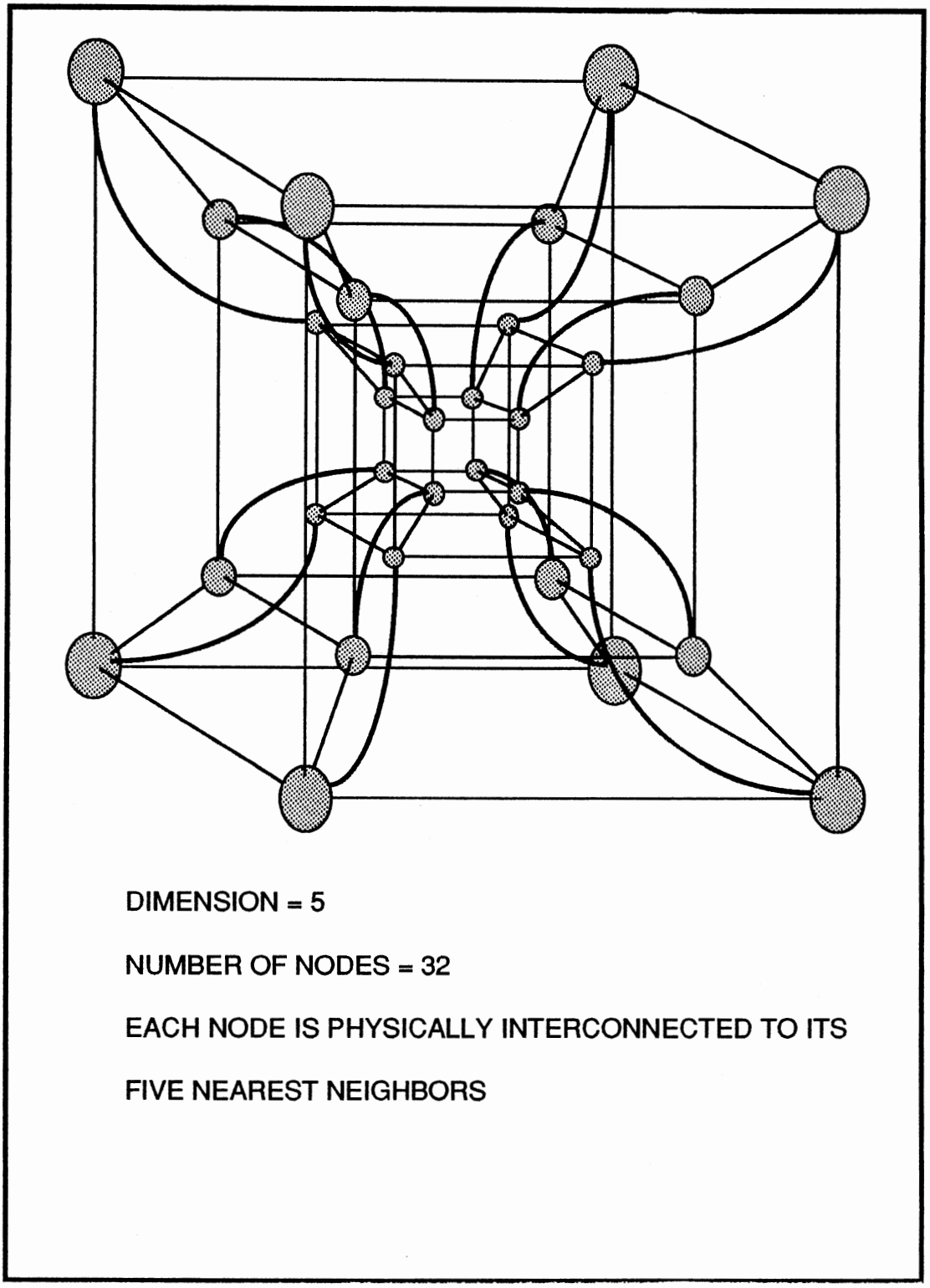


Figure 4. Hypercube Node Interconnectivity

operate on a problem in the parallel mode and show a dramatic speedup of the solution if the problem is appropriate and the solution is optimally programmed (Asbury , 1985). Details on traditional organizations and definitions of parallel processing parameters will be covered in Chapter III.

At this point, it can be recognized from the discussion that the solution for the equation in Figure 3 requires linear algebra investigation techniques that demand large numbers of simple independent computations. These can be dealt with as parallel primitives for concurrent solution speedup as suggested by the many authors of the DOA procedures (Bond, 1987). The objective here is to actually implement the procedures on a modern parallel processor.

It is important to note that this research goes a step further and reaches an additional level of speedup using multiple algorithms that are concurrently operating and cooperating within a single parallel computer. It will be seen that this second level of speedup is only available within specially organized parallel computers.

Dissertation Summary

This dissertation considers the application of a multi-processor computer system, using cooperating multi-algorithms, to develop a *signal-noise subspace* method of direction-of-arrival estimation in a multi-source environment.

The main driving factors are that the computations associated with the formulation of the sample covariance matrix, the eigenstructure analysis burden, and the subspace searches for linear combinations of steering vectors can be impractical in real-time situations for large antenna arrays on a serial computer.

One level of improvement is obtained by applying a parallel processor with P processors which can reduce the processing time of large array systems to approach an order of P less than that on the serial processor, provided P does not exceed certain limits compared to the size of the array.

Also greatly reducing the computational time, the typical N-cubed order eigenanalysis solution has been replaced with an iterative N-squared order procedure that also has a more favorable structure for parallelization. In this operation, using the acceleration technique of cooperation between multi-algorithms within multi-processors, the time required is again significantly lowered, allowing real-time solutions to be achieved on rather large antenna arrays.

The final level of development is in a new DOA function which integrates both the maximum signal and the minimum noise eigenvector components into a single estimator. The new function, which is faster to compute than other high resolution peaking functions, also enhances the closely spaced multiple signal resolution capability and increases the peakness of the output without affecting bias. It simultaneously lowers incorrect harmonic DOA bearing dominance due to sidelobe effects.

The organization of the remaining chapters is as follows. The specifics of eigenanalysis based DOA techniques are presented in Chapter II pointing out the four tasks that make up the heavy computational burdens, highlighting the parallel processing opportunities, and analyzing the MUSIC DOA method.

Chapter III is a discussion of parallel processing considerations including organization, parallel performance parameters, techniques and the modeling procedures that are followed. First level parallel processing is discussed showing capabilities and limitations. A new parallel technique considered in this dissertation to be at the second, and higher, level of parallelization called

cooperative multi-algorithmic acceleration is laid out in Chapter III for later chapter implementation.

The next four chapters are based on the four tasks identified in Chapter II and the parallel procedures edified in Chapter III. The parallel processing techniques and new approaches are combined with new algorithms to reach the goal stated in this chapter. It will be seen that as each chapter progresses, new serial and parallel algorithms are born out of the objective of real-time performance.

In these four chapters, each task will be mathematically approximated to assist in designing the algorithms and predicting the performance of the new algorithms when implemented in a multi-processor configuration. The DOA timing data resulting from computer simulation driven experiments is provided as a table at the end of each chapter. Comparisons can be made of each of the new parallel algorithms in terms of computation speed under controlled conditions. Timing considerations are the primary focus in these chapters.

Chapter VIII presents an ensemble model to assist in investigating final overall parallel timing optimization considerations. The overall timing table is compared to the overall mathematical model at this point to measure the agreement between the actual and predicted.

Chapter IX provides the performance of the estimator outputs of the developed parallel DOA procedure. This is partially accomplished by the same computer driven simulation procedures used for the timing data. Further, however, data obtained from an actual radio experiment has been included to collaborate the simulations.

Chapter X completes the dissertation with a summary and recommended areas of future research.

Contributions

There are many new results reported.

1) There is an improved algorithm swiftness in the serial mode, which is derived from utilizing a mixture of unique approaches. This is a result of an integration of the mathematical techniques available and is noteworthy in itself.

2) There is dramatic speedup performance which is a result of applying the first level of parallelization required new models and analysis procedures to be employed within the new hypercube parallel computer organization.

3) The multi-algorithmic acceleration procedure found in the dissertation is in concept and application a revision and a verifiable demonstration of an advancement beyond the previous constraints related to traditional parallel conversions of serial algorithms.

4) Although the signal subspace approach has been maturing for years, the closely spaced multiple signal resolution capability of the innovative high speed algorithm developed in this research shows very little estimator performance penalty compared to others presently available.

5) There is a unique procedure developed to estimate the number of arriving wavefronts based on the speed of convergence of the power method eigendecomposition while operating in a multi-algorithmic mode.

6) The integration of all of these elements have resulted in the outgrowth of a DOA algorithm that has super resolution capability, orders of magnitude speed improvement, and an eigenanalysis DOA technique that enters the physical real world with on-line solutions.

CHAPTER II

EIGENANALYSIS-BASED DIRECTION FINDING

There are a number of DOA array processing algorithms proposed in the literature that use the eigenstructure of the array spatial covariance matrix. DOA techniques are also valuable in digital spectrum analyzers for identifying sinusoids that are present, although they are not true power spectrum density estimators (Marple, 1987). The eigenvalue/eigenvector(EV/EV) based methods have considerable value due to their super resolution unbiased estimation capacities. This class of estimators developed by Pisarenko (1973), Reddi (1979), Schmidt (1981), Wax, Shan and Kailath (1982), and others make use of the eigenvectors associated with the eigenvalues of the spatial covariance matrix. The smallest eigenvalues are called the *noise eigenvalues*, and the largest eigenvalues are called the *signal eigenvalues*. The eigenvectors associated with the eigenvalues are a basis for either the noise subspace or the signal subspace respectively. Separation of the noise eigenvalues from the signal eigenvalues is one of the most difficult parts of the DOA EV/EV problem. The subspace separation is critical because it can be shown that the noise eigenvectors are orthogonal to the signal vectors, and is a basic element in the MUSIC approach (Kriel, 1988).

The conventional algorithms for DOA using the Fourier, maximum-likelihood, or linear predictive methods, although simpler in concept usually have poorer resolution given the same aperture size. This makes them less appropriate for

use when signals from closely spaced multiple sources are arriving at the receive antenna array (Reddi, 1979, Kriel, 1988). This is especially true in low SNR environments such as passive sonar (Johnson, 1982).

Results similar to the conventional algorithms can be obtained when using a DOA solution applying only the eigenvectors associated with the largest eigenvalues (Reddi, 1979, Johnson, 1982). This is similar to the beamforming approach that maximizes the signal correlation.

As was stated earlier, the improved resolution above does not come without a significant additional burden. Hence, elimination or reduction of the burden of extended processing time necessary to resolve the eigendecomposition DOA calculations is the motivation for the development of a multi-processor, multi-algorithmic accelerated procedure. The computation of the sample covariance matrix, its eigenvalue and eigenvector decomposition, and the large numbers of dot products required to compute what is referred to as the DOA spectra, are the bottlenecks in the real-time implementation of these high performance direction finders (Schmidt, 1981). As an initial solution to this problem, the bottlenecks can be assaulted with parallel techniques (Reilly, 1987). It will be found, however, to attain real-time DOA with large arrays, additional serial algorithmic advancements are necessary.

First an understanding of the eigenvalue techniques needs to be addressed. As a basis for later chapter developments the MUSIC algorithm will be the target procedure to generally illustrate the eigenanalysis techniques.

MUSIC Direction-of-Arrival

The following discussion will pin-point exactly where the large numbers of computations are required in an actual implementation of the MUSIC algorithm.

Typical solutions will be considered and improvements will be suggested.

The equation that defines the emitter localization problem is Equation (1-1). The **A** matrix, $\underline{x}$, $\underline{f}$, and $\underline{w}$ vectors were defined in Chapter I. This development will consider an equally spaced colinear omnidirectional antenna array as was illustrated as Figure 1 in Chapter I. The implimentation of the later chapters is also accomplished on a similar array.

In his 1981 dissertation R. O. Schmidt summarized the steps for the MUSIC algorithm to arrive at the angle of arrival for the large N, or the data "rich" case, with the following four steps:

Step 1: Collect Data to Compute Eigenstructure, **E**.

Step 2: Estimate Number of Signals, M

Step 3: Estimate Signal Subspace to Derive the Noise Subspace.

Step 4: Estimate Intersections of Signal Subspace with Array Manifold.

The analysis that follows will break down these steps into four tasks whose operations are the major computational burdens of the algorithm.

The first step of MUSIC, and other EV/EV methods, is to compute the spatial covariance matrix estimate, $\mathbf{R}_X$, from the signal samples and then decompose it into its eigenstructure. The covariance matrix is defined as:

$$\mathbf{R}=E\{\underline{x}\,\underline{x}^H\}. \qquad (2\text{-}1)$$

As stated earlier, in practice what is actually available is the averaged value over the S (S equals the number of snapshots) N by N (N equals the number of antennas) matrices between each snapshot of sample data. Hence $\mathbf{R}_X$, the sample covariance matrix, will only be approximately equal to **R**, the spatial covariance matrix.

$$\mathbf{R}_X = (1/S)\,\mathbf{X}\,\mathbf{X}^H = (1/S)\sum_{i=1}^{S} \underline{x}_i\,\underline{x}_i^H \approx \mathbf{R} \qquad (2\text{-}2)$$

The added noise has been assumed to be independent of the incoming signals and independent between antennas. It is also assumed to have a zero mean, which yields,

$$\mathbf{R} = E\{\underline{x}\underline{x}^H\} = \mathbf{A}E\{\underline{f}\,\underline{f}^H\}\mathbf{A}^H + E\{\underline{w}\,\underline{w}^H\} = \mathbf{R}_S + \mathbf{R}_W \,. \tag{2-3}$$

Hence, the estimated covariance matrix, $\mathbf{R}_x$, will be approximately equal to the sum of $\mathbf{R}_S$ and $\lambda\mathbf{R}_b$ where $\mathbf{R}_W = \lambda\mathbf{R}_b$. This can be rewritten as a generalized eigenvalue problem between the matrix pair $(\mathbf{R}_x, \mathbf{R}_b)$, and since $\mathbf{R}_S$ must be singular, being dimension N but rank M (M being the number of arriving waves) that is less than N, it follows that $|(\mathbf{R}_x - \lambda\mathbf{R}_b)|$ approximates $|\mathbf{R}_S|$, which is approximately equal to **0.**

$$|(\mathbf{R}_x - \lambda\mathbf{R}_b)| \approx |\mathbf{R}_S| \approx \mathbf{0}. \tag{2-4}$$

Equation (2-4) is true for all of the N generalized eigenvalues, λ_i, for the matrix pair $(\mathbf{R}_x, \mathbf{R}_b)$.

Many eigenstructure methods require the additive sensor noise to be spatially white, i.e., equal power and uncorrelated between sensors. The problem of nonwhite noise that has a known covariance can be attacked by solving the generalized eigenvalue problem (Paulraj, 1986). Note that $\mathbf{R}_b = \mathbf{I}$, an identity matrix of rank N, when the noise is considered spatially white. MUSIC assumes uncorrelated noise sources but also assumes a Gaussian distribution. Hence an estimate of $\mathbf{R}_b$ must be available to solve the problem in accordance with the above steps.

In the case of an unknown noise field, if the sensor noise is assumed white incorrectly the degradation of the estimate will occur such as bias, lower resolution, etc. (Martin, 1984). Generally, however reasonable solutions can be found assuming spatially white noise situation. It is clear that this is more significant in the low SNR situations, however highly colored noise will also effect estimates. These problems are of varying concern depending on the

problem at hand. Assuming the ability to estimate the noise environment in some manner exists, allows progression to the MUSIC solution of the generalized eigenvalue problem. Hence, there must exist nontrivial solutions such that the associated eigenvectors and eigenvalues are a solution to

$$\mathbf{R}_x \underline{e}_n \approx \lambda_n \mathbf{R}_b \underline{e}_n \ ; \ n = 1, 2, \ldots, N. \tag{2-5}$$

The term *eigenset* will be used to reference the paired eigenvalue and its associated eigenvector, λ_n and $\underline{e}_n$ respectively. Normally an eigensystem refers to all of the eigensets of a matrix. If the λ_n found above is the minimum of the generalized eigenvalues, then the equation for $\mathbf{R}_x$ is,

$$\mathbf{R}_x \approx \mathbf{R}_s + \lambda_{min} \mathbf{R}_b \, . \tag{2-6}$$

Since the matrix $\mathbf{R}_s$ is singular and has N-M eigenvalues that are equal to zero, λ_{min} must also have the same algebraic multiplicity, N-M (Schmidt, 1981). This requires that if there are five antennas and four incoming plane waves, the algebraic multiplicity for the zero eigenvalues of $\mathbf{R}_s$ and the algebraic multiplicity of λ_{min} would both be one. If one wave was arriving then the algebraic multiplicity of the minimum eigenvalue would be four. The number of arriving wavefronts can be approximated by using an estimate of the algebraic multiplicity derived during the eigenanalysis (Schmidt, 1981). The number of signals resolvable is a function of the number of elements, the SNR, the number of samples, etc.(Bresler, 1986).

Because the actual minimum eigenvalues are only approximately equal to each other, and not actually equal to zero, it is not a trivial effort to determine the correct number of waves even after all of the eigenvalues approximations have been extracted in the eigenanalysis procedure. Schmidt (1981) suggests a χ^2-based likelihood ratio test to determine the number of arriving wavefronts. Marple (1987) recommends the Akaike Information Criterion (AIC) as modified by Wax and Kailath (1985). Both of these methods requires the complete set of

eigenvalues to complete the calculation which leads to the estimate of M. Of course, both of these procedures are only estimates, so there is no guarantee that either method will produce an accurate count in a given situation. Some researchers have simply assumed the actual number of arriving waves as being a known quantity. They then focus on other considerations of the DOA estimation problem (Johnson, 1982, Reilly, 1987, Kriel, 1988).

Whatever method chosen, the number of arriving wavefronts is necessary information for accurate operation of the MUSIC procedure, and is the second step listed above. If too large an estimate for M is made, then extra peaks will be produced where no incoming signals actually exist. Likewise, if too small of an estimate is made, then arriving waves will be missed. This is effectively saying that, the estimate of the number of incoming wavefronts will normally be approached in the solution, correct or not (Johnson, 1982, Kriel, 1988).

An alternative method of estimating the number of arriving waves which does not require the complete eigensystem of the sample covariance matrix has been developed in this research and this contribution will be discussed in Chapter V. For the present, it is sufficient to assume that a value for M, the correct number of arriving signals, has been estimated using the MUSIC $\chi 2$-based recommended method requiring the eigensystem decomposition. Then it follows that:

$\mathbf{R}_S = \mathbf{A}E\{\underline{f}\,\underline{f}^H\}\mathbf{A}^H$, see Equation (2-3). And from Equation (2-5) it further follows,

$$\mathbf{A}E\{\underline{f}\,\underline{f}^H\}\mathbf{A}^H\,\underline{e}_n = (\lambda_n - \lambda_{min})\mathbf{R}_b\underline{e}_n. \qquad (2\text{-}7)$$

Thus $\mathbf{A}E\{\underline{f}\,\underline{f}^H\}\mathbf{A}^H\,\underline{e}_n = \underline{0}$ for the minimum eigenvectors associated with λ_{min}. But since $\mathbf{A}$ and $E\{\underline{f}\,\underline{f}^H\}$ are of full rank by definition, it is clear that:

$$\mathbf{A}^H\underline{e}_n = \underline{0}, \qquad (2\text{-}8)$$

for the eigenvector(s) associated with λ_{min}.

The "noise subspace" eigenvectors are the eigenvectors associated in

Equation (2-8), and the "signal subspace" eigenvectors are those that are remaining. The noise subspace eigenvectors are orthogonal to the signal subspace eigenvectors (Kriel, 1988). This results in the estimate of the signal nullspace (which can be considered the orthogonal complement to the signal subspace) as $\mathbf{E} = [\underline{e}_{M+1}, \underline{e}_{M+2}, \ldots, \underline{e}_{N}]$, and completes MUSIC step three.

MUSIC and other eigenvalue estimators use this information later to directly estimate the DOA. The solutions found, see Equation (2-5), to the generalized eigenvalue problem, see Equation (2-4), is one of the larger computational burdens in the MUSIC algorithm. The eigensystem decomposition usually requires an N-cubed order of computations.

In step 4 of the MUSIC algorithm, the noise subspace basis vectors are used by forming a function composed of the intersection with the array manifold, **A**, and matrix $\mathbf{E} = [\underline{e}_{M+1}, \underline{e}_{M+2}, \ldots, \underline{e}_{N}]$. Since the θ_m values in the $\underline{a}(\theta)$ elements are unknown, it is required to sweep θ through all possible arrival directions. The function $f(\theta)$, Equation (2-9) below, is sometimes called the peaking function of MUSIC because it will peak at the incoming angles θ_m.

$$f(\theta) = \frac{1}{\underline{a}^{H}(\theta)\, \mathbf{E}\, \mathbf{E}^{H}\, \underline{a}(\theta)} . \qquad (2\text{-}9)$$

In review, the matrix $\mathbf{E} = [\underline{e}_{M+1}, \underline{e}_{M+2}, \ldots, \underline{e}_{N}]$ is formed from all of the noise eigenvectors (remember there are an estimated N-M of these). The $\underline{a}(\theta)$ vector is a single column of the **A** matrix developed from the geometry of the array matrix (the stored array manifold) stated in terms of the arriving angle, θ. The estimation of the azimuth angles for the directions of incoming plane waves will be where Equation (2-9) peaks sharply as θ is swept from $-\pi/2$ to $\pi/2$ radians when using a colinear array as in Figure 1. In theory, without noise and round off error, $f(\theta)$ will equal infinity for values of θ that corresponds to the

actual directions-of-arrival of the incoming waves ($\theta = \theta_1, \theta_2, \ldots, \theta_M$). In practice due to the noise, limited sample size and finite computer word length, the function has a very large magnitude relative to those θ_i choices that are not close to a correct DOA.

This research recognized two important modifications that can be made to this function, Equation (2-9). They both improve on the speed of computation of the DOA process. They cannot be found by inspection, and will be generated as a result of the eigenanalysis in Chapter V and the intersection of the array manifold in Chapter VI.

Parallel Processing Opportunities

As can readily be observed, there are several areas that have the possibility of improvement by applying parallel algorithms to the solution. The specific computational burdens of MUSIC and other similar EV/EV procedures are divided among the following four operations:

Task 1: computing the sample covariance matrix,

Task 2: the eigenstructure analysis,

Task 3: forming the DOA spectra via the vector dot products, and

Task 4: locating the peaks in the DOA spectra (Schmidt, 1981).

Each of these tasks will be analyzed in detail in Chapters IV, V, VI, and VII, respectively. At this point a quick look will help size the problem as it has been approached with MUSIC.

The first operation is to compute the outer products to estimate the covariance matrix. This roughly requires the order of SN^2 calculations. As S approaches N this is essentially an N-cubed order calculation, after which it becomes a multiple N-cubed order task.

In the second operation, MUSIC requires the entire set of eigenvalues to be evaluated to estimate M, the number of incoming signals. The more arriving signals, and the lower the SNR, the harder it is to differentiate between the noise subspace and the signal subspace. With few arriving signals, larger numbers of noise eigenvectors need to be resolved. Using traditional complete eigendecomposition will require extensive time for this operation which is typically of order N-cubed (Tufts, 1986).

The third operation of MUSIC is a function of the number of incoming waves and requires (N-M) times the N-order dot product for the computations for each angle (or portion of an angle) investigated. For small M, and assuming at least one tenth of a degree resolution for the angle, this function will also be of the order of multiple N-cubed computations.

The last operation is the search through the computed DOA spectra looking for the peaks. Knowledge of the number of arriving waves from the second task determines the number of peaks sought in this operation. This search will also be a function of the total number of azimuth bearings in the search, the resolution of the search, and the number of signals expected to be arriving. It is not a function of S or N, but can be a large time consumer when the brute force method is used for noncolinear antenna arrays (Speiser, 1985). For small values of S and N the time for the first two tasks diminish and this as well as the third operation represents a fixed time cost base on the number of bearings investigated.

Although this study is a two dimensional azimuth-only direction finding procedure, when a three dimensional vector is used as a result of other than a colinear array geometry, then each bearing of azimuth must be completed for each angle of elevation investigated. This would have the effect of increasing the search time significantly causing this task, and the last one to have an

increase in the computer time used.

The mathematical simplicity of these dominating tasks and the opportunity for multi-algorithmic application in the eigenstructure analysis, makes MUSIC or EV/EV variations of it, top candidates for a parallel processor applications.

Although all actual computer times will be presented, and they can be placed against any criterion desired, for simple reference, the goal of real-time, or on-line, processing for this problem will now be defined. Often relative values of improvement are used to establish what is meant by real-time performance (Bromley, 1985). That is, if the speedup improvement is of the order 10N, where N is the primary dimension of the system, then this might be referred to as being real-time (Reilly, 1985). An improved definition of real-time performance can be based on input, control and response. In this environment, real-time is as long as the result is available to a continuing problem for a particular set of input values while those inputs can be affected (Morris, 1977). But of course this might allow a delay ranging from microseconds to hours, and still be labeled real-time.

Keeping these ideas in mind, but desiring to be a little more specific, an absolute time value has been selected to define real-time performance in this research. Somewhat arbitrarily, but within practical reason, if the output is obtained in two seconds or less, then the goal of real-time processing will be considered to have been met. Any time of two seconds to ten seconds will be defined as being *near real-time*. These definitions are not provided to diminish results that show above ten of seconds of processing time as opposed to many tens of minutes or even an hour or more required in previous serial DOA systems. These kinds of results simply fall out of the real-time and near real-time definitions provided above, but will certainly be sought after and considered valuable.

Clearly real-time processing is dependent on the application. Situations can easily be constructed when two seconds is not a reasonable solution time or when two hours is more than fast enough. These times have been chosen considering normal applications in the sonar and radar arenas. Constraints do result with the establishment of these two windows, as they provide a threshold for desired performance. The limits that these definitions place on the problem size can be more clearly understood when a function analysis is placed against a typical time line.

Figure 5 compares the impact of X, Xlog(X+1), X^2, and X^3 functions against a time line (in a log scale) for increasing values of X. It can be seen, that with X at 712, computer time required using a .1 megaflop machine exceeds 1 hour in the X^3 function case. Hence, even a single X-cubed factor makes it virtually impossible to improve an algorithm to real-time by applying only 16 processors in a straight forward parallelization as X increases to above 150.

It has already been pointed out that this problem has a large number of multiple N-cubed factors, without even considering the timing effects of the actual coding implementation. To further extend this problem, the solution requires the use of complex mathematical representations causing even larger numbers of computations. This implies that straight parallelization of multiple loops in these DOA algorithms could not be successful except for small array sizes. It will be discovered in the next chapter that when the message passing parameters in a concurrent solution are also a function of X, it is typically not possible to apply hundreds of parallel processors when the problem factor, vector length or matrix size for example, is also only into the hundreds. This limit will be called reaching the first level *parallelization threshold.* In fact, the order of tens of processors will be found to provide the optimum speedup parameters for this size problem. Sixteen processors are the maximum level

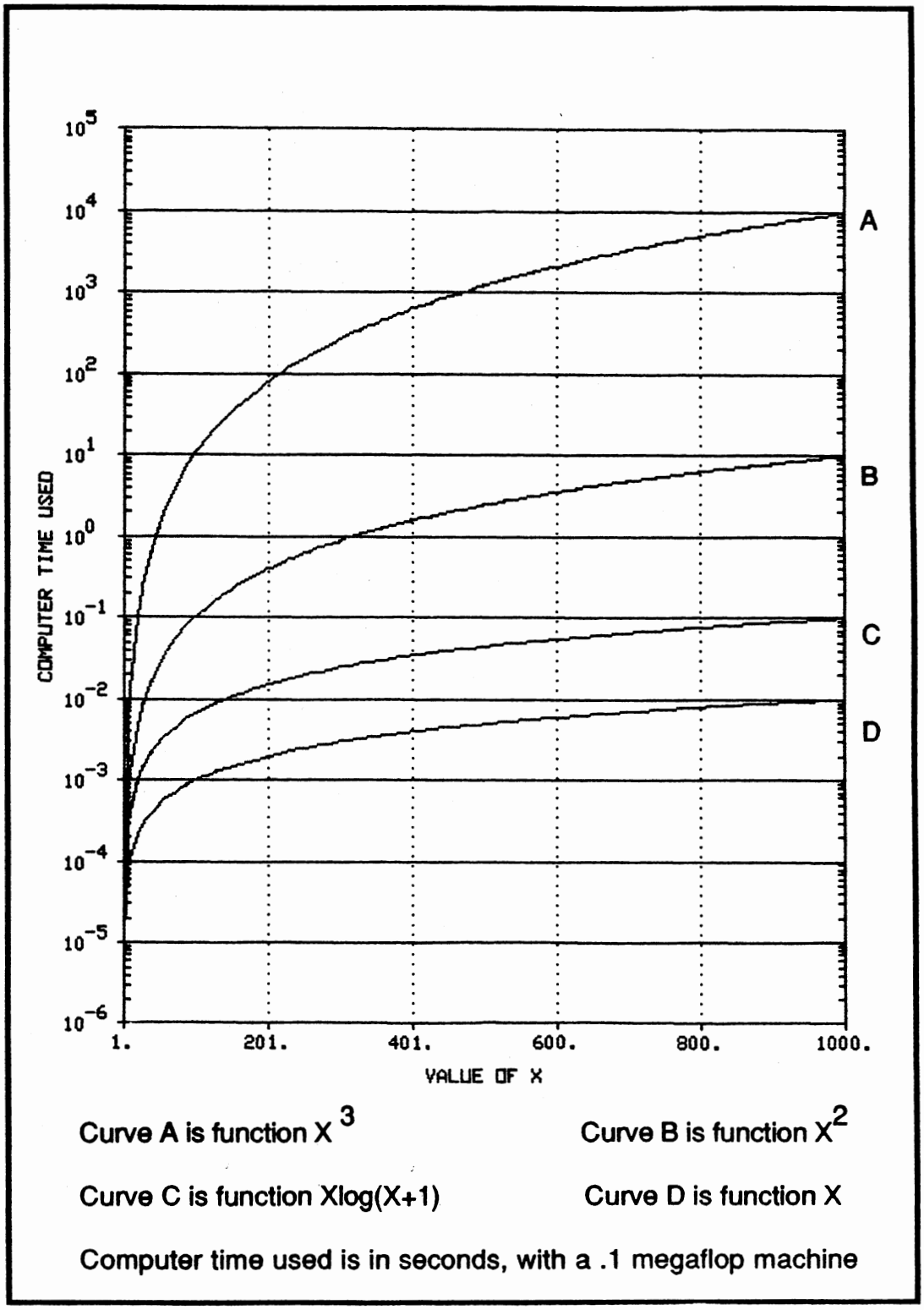


Figure 5. Increasing Order of Various Functions of X

typically applied in this research.

Hence, the only way to reach a real-time processing goal without resorting to supercomputering speeds, is to reduce the algorithms to N-squared order procedures. Then, with the application of tens of processors, the computer time will become less than two seconds, even for large arrays. Of course, very large arrays can still be processed if memory limitations do not constrain the processing capabilities, and dramatic time savings can be obtained.

Figure 5 also shows that problems with functions of order Xlog(X+1) and X will yield real-time speedup without resorting to parallel processing speedup modifications. The actual time improvment using parallel processing in these cases could not be justified unless the size of X extends past ten thousand.

A better understanding of the limitations and advantages of parallel processing improvement will be addressed in the following chapter along with the measures of parallel performance and parallel computer organization.

CHAPTER III

PARALLEL PROCESSING

Solutions to problems that require manipulation of large data matrices and vectors such as the computation of the sample covariance matrix and the eigensystem decomposition as discussed in Chapter II, or any problem that has nonunique solutions, such as array signal processing analysis, can reasonably be expected to be resolved in less time with parallel processing (Huang, 1980). In applications where iterative solutions are applied, parallel processing with multiple algorithms can often provide an additional reduction in processing time. With these kinds of large problems it is often possible to save *significant computer time* with the application of a parallel processor system.

This chapter examines the parallel implementation techniques and parallel algorithmic modifications to reach this research's goal, a multi-processor, multi-algorithmic accelerated, high resolution DOA real-time estimator.

The beginning of any parallel procedure lies with the problem being solved. The previous serial results, if any, are an extremely important consideration. Even though the parallel algorithm may require an entirely new parallel solution development, research of these previous serial solutions is done to uncover the types of parallel opportunities such as matrix-vector multiplications and vector or matrix manipulations that exist within the many completed works. Parallel mathematical primitives can be developed based on the parallel architecture of the parallel processor being applied and the problem being

solved. This will guide the parallel design to the best algorithm for parallel implementation. As the approach is expanded, and new and old methods are judiciously considered, a new-born optimized parallel algorithm results.

Attention should be called to the fact that the most efficient serial program may not have much in common with the most efficient parallel algorithm. In efforts of serial optimization, the serial algorithms frequently take on a very special strictly serial nature that does not yield as much improvement through parallelization. The use of singular value decomposition (SVD) in the serial eigenvalue program for MUSIC is usually considered the optimized serial method (Luk, 1987). It would be wrong to immediately extend this answer to a parallel beamforming algorithm without attempting a fresh parallel approach.

In the case of this research, even though the initial focus was on the MUSIC DOA solution, what follows is a new serial algorithm, a new parallel algorithm including new a parallel approach, and what must be considered a different and improved solution to the multiple signal DOA problem.

Chapter II listed the four tasks that comprise the computational efforts in typical eigenanalysis solutions. It was stated that other than being of multiple N-cubed order, these tasks have the ideal setup for development into an efficient concurrent solution. Chapter IV will begin the specific task analysis and alteration of these tasks. This chapter, however, will first detail some parallel computer organizations, discuss the parallel performance parameters, develop the advanced parallel feature of multi-algorithmic acceleration, and present the mathematical model to be followed.

Parallel Computer Organization

Practical parallel processor hardware falls into two major categories. As

depicted in Figure 6, parallel processors are usually either single-instruction stream multiple-data stream (SIMD), or multiple-instruction stream multiple-data stream (MIMD) (Flynn, 1972). The SIMD machine has multiple processors operating with the same instruction stream upon multiple sets of data. The most common SIMD implementations are systolic algorithms on systolic array hardware as found in special-purpose applications such as image processing. Systolic array systems operate concurrently and sychronously on the data using a parallel arrangement of pipelined processors to yield very high speed parallel solutions. One big benefit of this structure is that the final systolic design applications readily lend themselves to VLSI hardware implementation (Bond, 1987).

The MIMD organizational structure has an advantage over SIMD in that it can emulate the systolic array system (or many other architectures) for prototyping or in actual operation, but unlike SIMD machines, MIMD machines can also operate simultaneously with entirely different algorithms at each processor using entirely different data sets. This is the origin of the advanced MIMD parallel computing power associated with multi-algorithmic acceleration which is discussed at the end of this chapter.

Within the category of MIMD structure, two primary methods of passing data between processors have evolved. The processors either share memory connected by a common hardware memory bus, requiring some method of memory access arbitration, or they send explicit messages over a communications network between the processors, requiring a method of message routing (Karp, 1987).

Timing data favors shared memory methods because of the narrow communication bandwidth's available for message passing compared to the latest high speed memory access times. Additional significant time delays

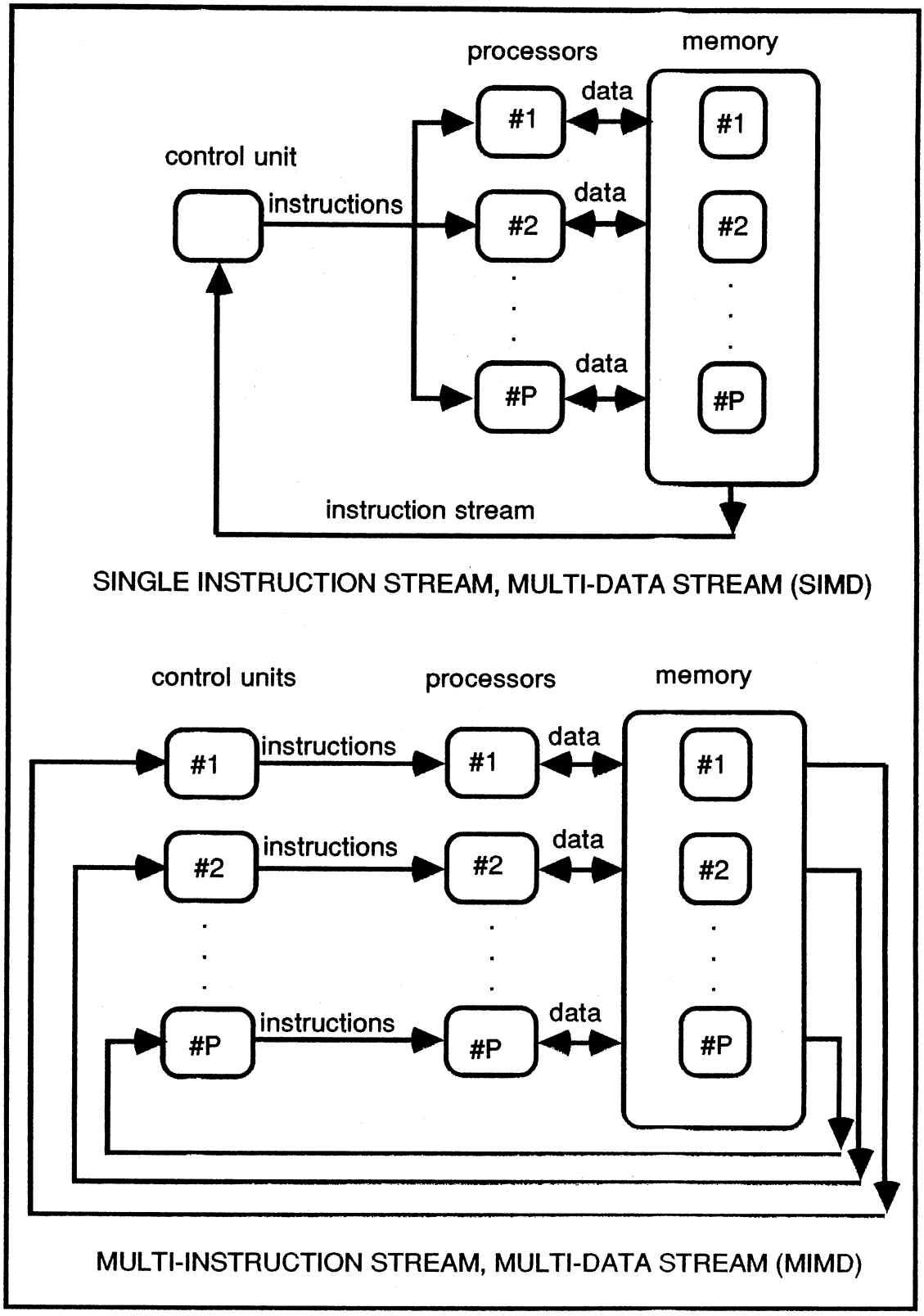


Figure 6. Two Practical Parallel Processor Organizations

occur in many message passing systems because store and forward message routing techniques are often used between the cascaded connections of processors. Of course, the shared memory organization is not unbounded and it has some unique disadvantages compared to the message processing organization. Limits on expandability of the bus structure to access memory, increasing conflicts on memory and bus arbitration, and software timing difficulties are some of the shared memory computer organization basic weaknesses (Flynn, 1966).

The Intel iPSC/2, a message passing hypercube organized computer, has narrowed the timing gap with it's new node to node communication technique by using what is called a Direct Connect Module that allows internodal messages to be sent virtually from any node to any other node at a data rate of 2.8 MBytes per second.

Figure 4 shown earlier is actually the physical nodal connectivity of the iPSC/2 MIMD hardware. Research was completed in the configuration of a hypercube using only 16 nodes. This organization would be defined as a cube of dimension 4 (where 2^4=16 nodes). Prior to current routing schemes, computers organized this way caused algorithms being produced to include very careful analysis to assure minimum routing conflicts over the limited interconnectivity. However, the Intel computer's newest message routing and message switching hardware scheme makes it appear as if it were an ensemble of fully interconnected processors (Intel, 1987).

Even with these recent advances, transit time of node to node messages is still a major parallel software design concern. The ratio of calculation time to communication time for the nodes is a driving factor in maintaining or reaching optimum speed, hence it greatly affects parallel computer performance and speedup parameters (Grunwald, 1986).

Compared to a serial machine however, whichever parallel configuration used, when the nature of the problem is as stated above, dramatic speedup can usually be achieved at a much lower cost (Ratter, 1985).

Parallel Computing Performance Parameters

The reference of saving "significant computer time" in this chapter's introductory remarks was an intentional highlight relating to a weakness in the current measure of parallel speedup performance. Logically, speedup is simply the time required on a serial computer compared to the time required on the parallel computer. However, the term speedup normally refers only to a ratio of these two numbers (Schendel, 1984). Because it is then unitless, It overlooks the absolute amount of time saved or used by the parallel processor.

To overcome this problem, two parameters, S_ρ= *speedup ratio* and S_δ = *speedup differential*, are defined below as functions of parallel performance, and the term S=*speedup* will generally be used as meaning both. The symbol S has already been used as the number of samples, but no confusion should arise from these quite distinct assignments.

The following example is provided to see the impact of having only a single speedup definition. Suppose an optimised serial algorithm takes .001 seconds to solve a problem. Then, with the application of a parallel computer with 128 processors, assume that the parallel system can solve the problem in only .00001 seconds. This would yield a speedup ratio of 100.

$$\text{Speedup Ratio, } S_\rho = \text{serial time/parallel time}$$

$$.001 \text{ sec.} / .00001 \text{ sec.} = 100 \qquad (3\text{-}1)$$

Reporting an overall speedup of 100 has the ring of a very good result. On the other hand, reporting the total speedup differential improvement of .00099

seconds will take some tall justification to be called a valid parallel processor application.

Speedup Differential, S_δ=serial time - parallel time

.001 sec. - .00001 sec. = .00099 sec. (3-2)

Here it is important to chose a problem that consumes impacting amounts of computer time before the expense of a parallel processor can be justified.

The other end of this problem would be to report a speedup differential improvement of 10 minutes, but not pointing out that the speedup ratio is 1.1 and it takes 100 minutes of computing time to achieve the improvement. This time, the problem seems to warrant application due to the large computer time used, but the parallel improvement is probably not one that is reasonable to consider a parallel processor application.

This situation is analogous to trying to use only an absolute error value or only a relative error value in approximation problems to measure success. In that regard, any parallel improvement will be described by both of the speedup measures defined above to provide realistic and practical dimensions of the improvement.

Another performance measure is *parallel system efficiency*. The traditional definition of percent efficiency, or simply efficiency, has the speedup ratio divided by the number of processors, times 100 (Fox, 1987). This method assumes the best, or 100 percent improvement, occurs when the speedup ratio equals to the number of processors. Another way of looking at this definition is that the best that is expected with P processors, is to reduce the time to 1/Pth the serial time. The example above with 128 processors and a speedup of 100, yielded a parallel processing efficiency of about 78 percent.

Efficiency, $E = (S_\rho/P) * 100\ \%$

$100 * 100$ (speedup ratio) / 128 (number of processors) $\cong 78\%$ (3-3)

An efficiency curve can be provided for a particular algorithm. The curve would plot the efficiency versus the number of processors applied to the problem. Efficiency is often mostly a result of the nature of the problem. Because of that, a strongly serial problem would not yield good efficiency performance even when small numbers of processors are applied. On the other hand, a highly parallel problem would very closely approach one hundred percent even with the crudest parallel design (Asbury, 1985). The efforts to provide data for efficiency have not been expended because with a simple observation of the speedup ratio data, efficiency can be quickly and accurately extracted. The closer the speedup ratio obtained comes to the number of processors applied, the closer the system will be to the ideal 100 percent efficiency.

Efficiency, being dependent only on the speedup ratio can also be misleading. This is the case when a very high efficiency can be attained, yet a less efficient parallel algorithm exists, that has much better speedup differential times. Obviously the best application would be the one that has the fastest solution, even if the efficiency and speedup ratio values are worse. Certain conditions cause the first operation developed in Chapter IV to present this exact event, and this situation will be further highlighted there.

Also, the fact that the system is running fifty to seventy percent efficient is not necessarily significant. This is not saying that efficiency is a useless parameter, it is just that the computing power lost is normally not available for other uses. Therefore, knowing that an inefficiency exists may not be a real world concern.

Comparing the speedup curves and efficiency against an increasing number of processors usually shows efficiency curves that have a peak at a less than absolute maximum speedup value. At that point, adding more processors may still improve the speedup parameters, but the efficiency begins to drop. Further

after that point, continuing to add processors, a point will be reached where adding more processors will degrade the resulting speedup parameters causing *speeddown*. This effect is called exceeding the *parallelization threshold*, and can be defined in terms of the speedup ratio as "*that point at which adding more processors to a parallel solution yields a decrease in the speedup ratio*". An equally correct definition using the speedup differential parameter would be stated as "*that point at which adding more processors to a parallel solution yields a decrease in the speedup differential*".

This situation is most often due to the increased communication delays compared to the decrease in computing time used by each processor as a result of adding the extra processors (Bond, 1987). In a later example, the parallelization threshold will clearly be exceeded, causing a speeddown. This problem also occurs in the implementation of the second task using small arrays. Chapter V contains the data and explains some of these results.

The speedup ratio is a direct function of the percentage of time each processor is used to do concurrent calculations. The overall time used for the algorithm is a function of the number of calculations being done concurrently, the amount of time spent running strictly serial code, and the amount of time spent communicating between processors. From this it can now be concluded as was stated earlier, that best parallel performance occurs when the ratio of the time spent doing concurrent calculations to the time spent in communication and within the serial mode is maximized.

First Level Multi-Processing Speedup

As was already discussed, parallel speedup is normally accomplished in the area of partitioning the matrix vector manipulations by spreading out the basic

inner loop multiplications and summations among the many processors allowing their concurrent computation. Timing improvement of this kind will be referred to as *first level* (parallel or multi-processing) speedup.

The following analogy illustrates the effects of the first level of parallelization speedup. This level is analogous to taking P processors and assigning the total group to directly sum (no mathematical short cuts) all of the integer numbers from 1 to X. A straight forward and logical approach would be to split the total X-1 adds into P, (X/P) -1 concurrent adds by each processor, to obtain P partial sums.

As a side note, notice that certain restrictions between the relationship of P and X needs to be maintained to assure the (X/P) term is always an integer. That is, in this example, it would not be possible to compute other than an integer number of calculations. When other than an integer results from the splitting process, this indicates that it is not possible to equally distribute the particular problem among the P processors. In this case, a subset of processors equal to the remainder in the division would require one addditional calculation compared to the other processors. In most cases, each processor's calculation workload will normally be into the millions, therefore this is normally not a negligible workload balancing problem.

Further, given the equal splitting situation, this case can be totaly ignored. In fact, throughout this work unless stated otherwise,it will be assumed that the input data can be uniformly split between the processors applied and no generality of the solution will be lost.

Continuing with the example, at least one supervisor level would need to exist, however this level could also do a portion of the sums between supervisory activities. The P partial sums would require $\log_2(P)$ more adds. Since dimension of the cube squared equals $\log_2(P)$, then the number of

messages required is the same as the dimension of the cube applied. Figure 7 illustrates the partial sum combination method that results in the $\log_2(P)$ additional sums to be required. Additionally then, each partial sum takes some finite amount time to be transmitted between nodes. The ratio of the minimum message communication time to a single compute time will be denoted, μ. Hence, $\mu\log_2(p)$ is the transmission time of the partial sums between nodes to reach the final answer. The speedup ratio parameter is represented by:

$$S_\rho = \frac{(X-1)}{((X/P)-1)+(\mu+1)\log_2(P)} \tag{3-4}$$

and the speedup differential parameter (reflecting the serial time saved) would be

$$S_\delta = (X-1) - (((X/P)-1)+(\mu+1)\log_2(P)) \tag{3-5}$$

multiplied by the time required for one floating point operation.

The quantity $((X/P)-1)+(\mu+1)\log_2(P)$ multiplied times the time for a single flop represents the computer time used in the parallel mode for different values of P and X. Using a constant value of 50 for μ, .00001 seconds for the time for a single calculation, Figures 8, 9, and 10 provide the speedup ratio, the speedup differential curve, and the computer time used (CTU) for this summation example problem. These curves are for values of X equal to 32; 1024; 32,768; and 1,048,576. The number of processors, P, ranges from 1 for a serial computer, to 31 for a dimension 5 hypercube less one node.

For larger and larger X, the speedup ratio, Equation (3-4), asymtotically approaches the value of P, making a straight line. Figure 8 indicates that with a very large X, the job can be completed in almost (but never equal to, of course) 1/Pth of the time it takes on a single serial processor.

For smaller values of X an important effect occurs. It can be seen that for X=32 speedup is always less than one, hence speeddown always occurs when

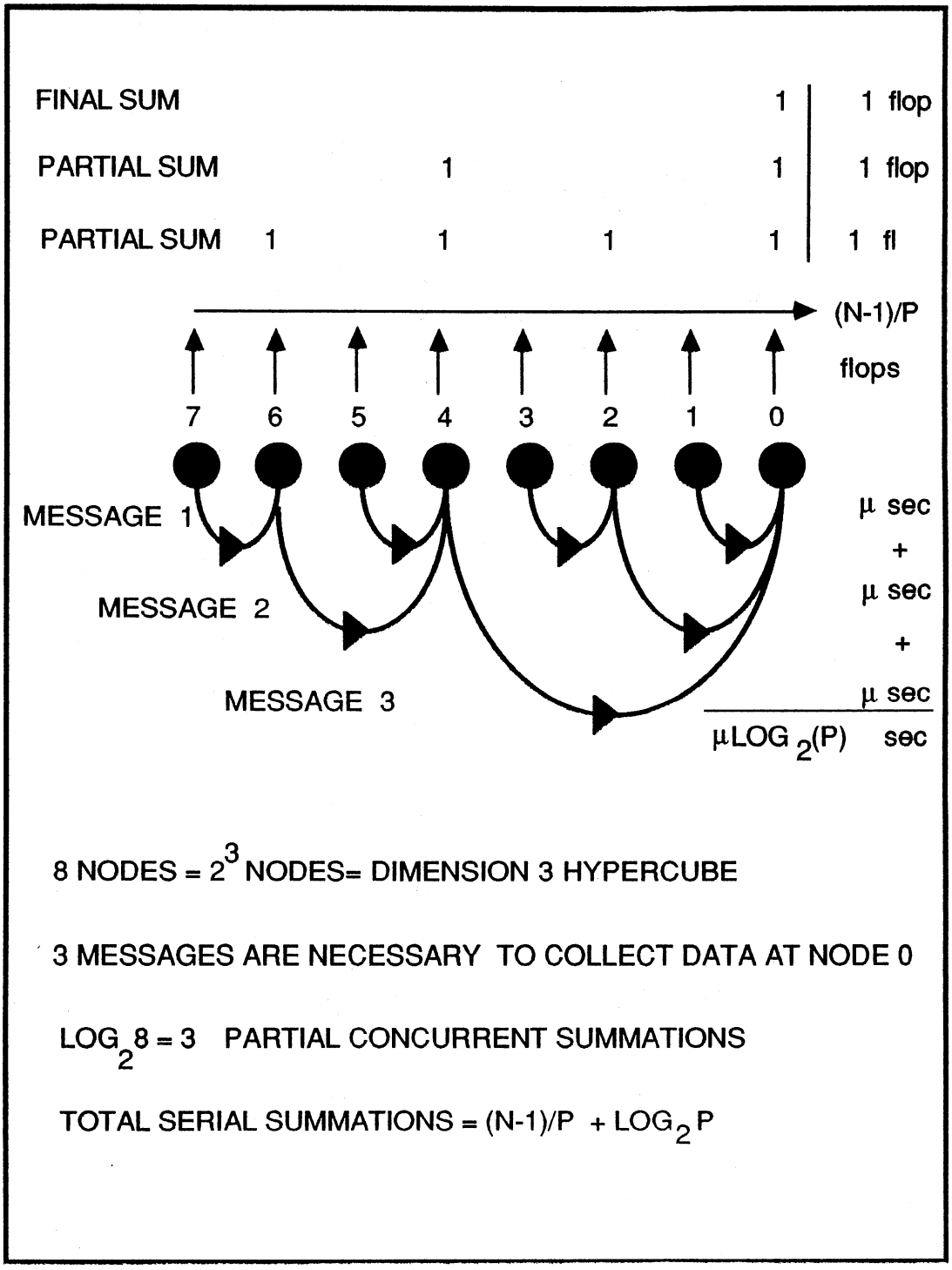


Figure 7. Summation and Message Increases as a Function of $Log_2(P)$

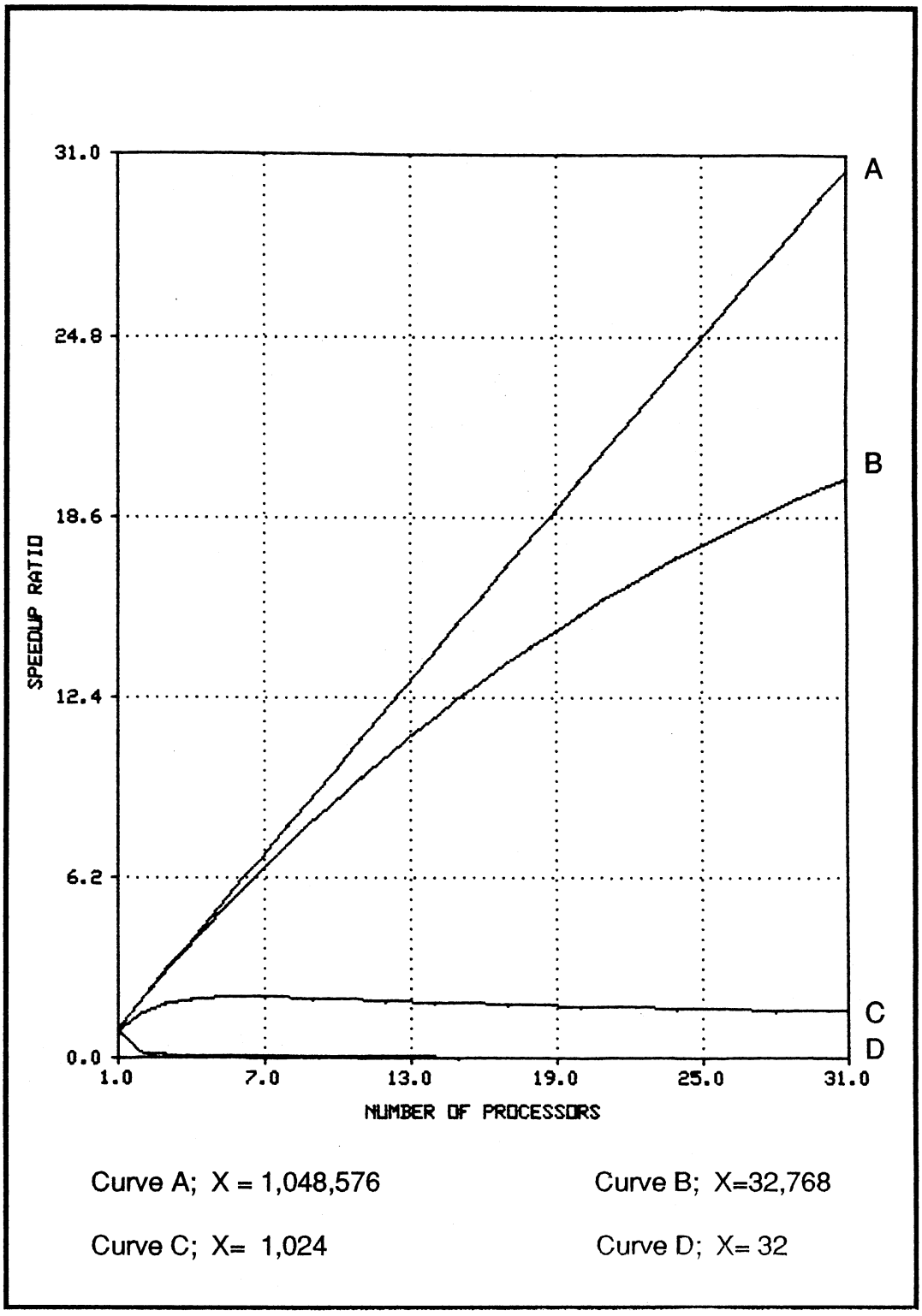


Figure 8. Speedup Ratio for Summation Problem

trying to apply parallel processing to a problem of this small size. Even for an X as large as 1024, when 7 processors are applied, the problem conditions exceed the parallelization threshold and the time consumed is larger than it took with only 6 processors.

For larger X values the threshold effect does not occur when using only a maximum P of 31 processors. However, a parallelization threshold exists for all values of X and can always be exceeded given enough processors. This clearly shows that there are always conditions that using more processors is not the correct decision.

The situation contributing to the problem in this example is that the message processing delays incurred when additional processors are included, are longer than the time saved by the computational improvement of adding more processors. Additional serial processing will further contribute to this problem, however it is not modeled in this particular example.

The computer time saved through parallelization is equal to the speedup differential, Figure 9. The computer time used (CTU) for varying values of P is given as Figure 10. The CTU plot is extremely valuable considering visibility of the absolute time change in performance due to parallelization of the problem. First, because it is always positive, a log scale for time can be used allowing a greater dynamic range of values when plotted. Second, when P is swept from one to the maximum value of processors applied, the value for one processor is simply the serial CTU. With direct speedup differential curves as the plots or data tables, with one processor this value is equal to zero indicating no time being saved.

Figures 9 and 10 also provide insight into what adding more processors means in terms of absolute time savings. It is shown that each doubling of the number of processors applied, can only save at best, an additional 1/Pth of the

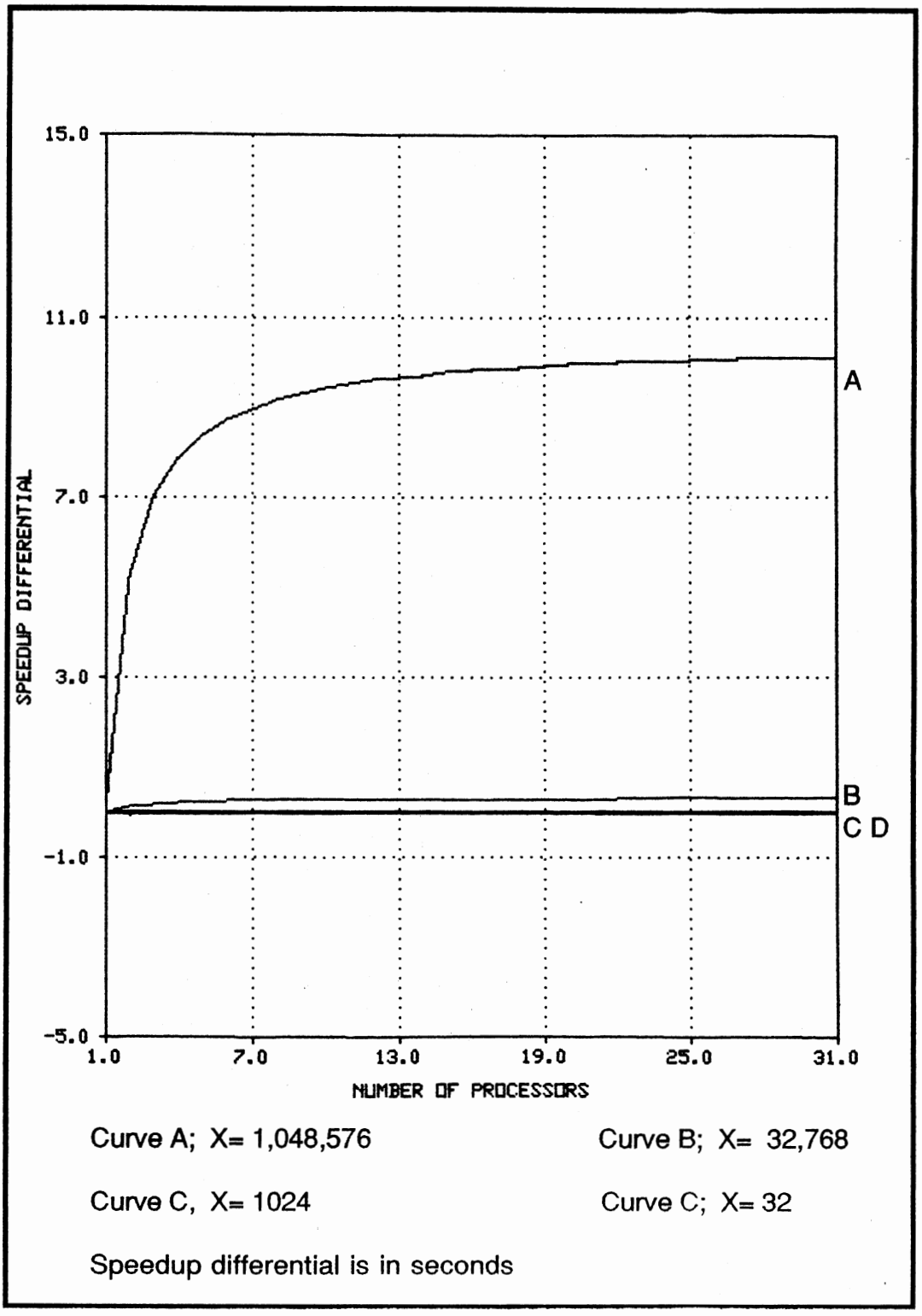


Figure 9. Speedup Differential for Summation Problem

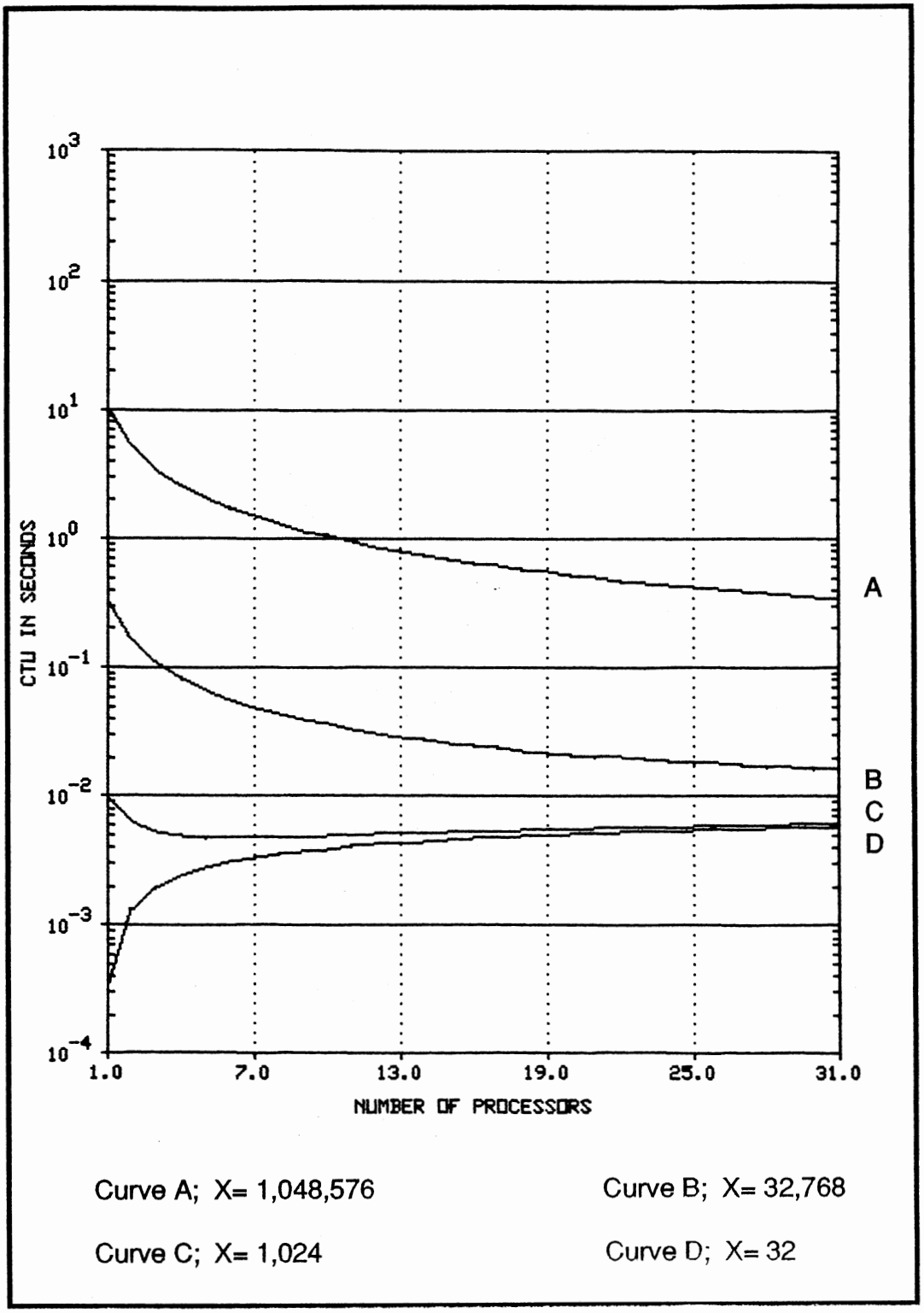


Figure 10. Computer Time Used (CTU) for Summation Problem

total time a serial computer requires. In other words, with four processors the most time that can be saved is 75 percent of the serial time. With eight processors at most 87.5 percent can be saved, so the improvement by adding four more processors was only half of the 25 percent that was left, or one eighth of the total serial time. Now, adding eight more processors, for a total of 16, can only save at the very most an additional six and one quarter percent, or 1/16 of the total serial time. This gain of 1/P percent for doubling the processors is not a function of this particular example, it is in fact the ideal that can be expected with perfect parallel processing.

Once again, since these figures are for the near ideal case, they illustrate the greatest first level speedup benefit that can be expected by simply adding more processors to a task. Of course, if the algorithm is written for an SIMD computer, then assuming the process does not speeddown due to exceeding the parallelization threshold, any gain is some gain. However, as will be shown, there is a more efficient simultaneous application of the extra computing power available when using a MIMD computer organization.

All three plots or the associated data tables are not necessary, because the same information can be easily obtained from any two. All later experiments will furnish tables with the speedup ratio, $S\rho$, and the computer time used, CTU, allowing efficiency, E, and speedup differential, $S\delta$, to be extracted by a simple observation whenever they are needed.

Before leaving this first example, there are a few key assumptions that were used that need to be stated that greatly affect this simple model.

It was assumed in the example that all the processors work at the same pace. If one or more of the processors has a capacity for greater speed, then this special ability should be exploited, further increasing the complexity of the *workload balance*, but necessary to keep all of the processors productive.

It was also assumed that because the processors are equal in computing speed, the workload balance was evenly distributed as long as it is possible to give an equal number of integers to each processor. Distributing the workload evenly may not be this simple of a matter. Even in this case where the adders (processors) were assumed identical and the number of integers are equally distributed, it could still be much quicker to add the integer numbers 1 and 2, then to add the integers 138,982,648,937 and 138,982,648,938. If this were the case, then the workload should be distributed to the nodes according to the number of digits in the integers, rather than the number of integers.

Next, it was assumed that the extra computations and manipulations necessary in the actual implemented code could be accomplished at a lower order compared to the original problem, hence inclusion of their effect can generally be neglected for larger problems.

Lastly, and more significantly, a constant time has been modeled to send the partial sums between processors. The data (messages) will clearly take some amount of time to pass between nodes. In many cases it can often become a major factor in the total time used when large N by N matrices need to be transported between nodes instead of a single data word.

Although this example was very basic and the assumptions somewhat fundamental, it was presented because it embodies some of the ideas and problems of the traditional parallelization efforts. It also provides quick insight into the parallel processing problems of speedup, load balancing, and efficiency.

This first level of speedup is roughly equivalent to using vector boards for matrix methods on serial computers and has proved worth-while. It is also a level of parallel processing that can now be partially accomplished by taking standard serial programs and compiling them with a parallel compiler for some

special purpose machines (Egan, 1988). In this case, only computational loops of the program would automatically be selected by the compiler for first level parallelization. This would limit the speedup performance possible, but improvement could be attained quite easily by the user. Much greater gains can be obtained with parallel algorithms designed for parallel processors.

The advantages the MIMD organization has over the SIMD organization is not used at this first level of parallel conversion. This is because a common approach is to cause each of the processors to run the identical algorithms, and only use different input data. This would caues the MIMD organization to perform basically the same as the SIMD organized computer. Instead of being called an MIMD implementation, it could be relabled an MIMD/single-program multiple-data (SPMD) approach.

Much work has been accomplished by using the MIMD machine in the SIMD and the MIMD/SPMD modes. This is because sequential algorithms have a developed base of many years of experience, hence their understanding is much greater than the directing and coordination involved in parallel computers (Denning, 1985). The problem is not that these modes are totally undesired, it is that often the extra-ordinary capacity of simultaneously running entirely different algorithms on the same or different data is completely ignored by the parallel algorithm software designer. This is often due to concerns of algorithm complexity, timing, portability and clarity of algorithms rather than lack of application (McBryan, 1987).

Cooperative Multi-Algorithmic Acceleration

This research is designed to reach the next higher level of speedup using the MIMD capabilities when several different algorithmic approaches, approach

variations, or different data are available for the same problem. With this method different algorithms are given a variety of starting points and can exchange information on a cooperative basis. This can shorten the problem solution time by allowing the algorithms to use the improved information to accelerate their individual positions interactively.

It will be helpful to describe the proposed procedure with a simple example as before. Now let the processors, still P of them, have a difficult problem. Let this problem have a nonlinear solution, and let different procedures exist to reach an estimated solution. Although the procedures may be different, they often have some common primitives and functions for they are solving the same basic problem. This could be related to the real situation of having many different methods available to solve eigenvalue problems, but vector-matrix and matrix-matrix primitives appear in most techniques. This commonality will be used as the first level parallelization improvement when possible.

Nexr, also assume these high level problem solvers can help each other as they progress by communicating interim results. This could be information about real or false paths determined, speed of algorithm convergence, residual error size, and other determined parameters.

If this problem has three known algorithms as solution procedures, then three sets of the processors would be used for each solution at the first level parallel improvement of work. Three processors would be controlling the operations as supervisors of the three different algorithms, causing interchange of data and interaction between the solutions. It is possible that this would mean a slow down from the highest speedup compared to a single parallel algorithm using all P processors at the first level. But as was shown earlier, this is not always a significant degradation, nor necessarily a slowdown at all. There can be improvement shown at both the first and second levels of

speedup when the optimum number of processors are applied.

It can be shown at this point that due to multi-algorithmic procedures, a weakness in the standard measure of parallel processing efficiency also exists. That definition assumed, the best that could ever be expected to be reached with P processors being applied, would be a speedup ratio of P, yielding 100 percent efficiency.

To demonstrate this weakness, consider a serial iterative algorithm that takes X seconds to solve a problem. Breaking down the work further, might show that it actually takes I times T seconds, where I represents the number of iterations required, and T is the time for one iteration. Applying a parallel processor at the first level by partitioning the problem into P elements, each taking 1/Pth of the time T, would yield an efficiency of

$$I\,T = X \text{ sec.}$$

$$I\,(T/P) = X/P \text{ sec.}$$

$$((\,X\,/\,(X/P)\,)\,/P\,)\;100 = 100 \text{ percent.} \qquad (3\text{-}7)$$

This is the best case possible obviously ignoring some of the real world problems allowing the process to reach the ideal 100 percent efficiency.

Starting with the same assumptions, let the total iterations that are needed be cut from I to I/4 during the computations due to information from two cooperating multi-algorithms (each running simultaneously on half of the processors). Then the time used, assuming the first level speedup procedure was still being applied but was half as effective because of the distributing of the processors (being the worst case that could result) :

$$I\,T = X \text{ sec.}$$

$$(I/4)\,(T/P/2) = I\,T\,/\,(2P) = X\,/\,(2P) \text{ sec.}$$

which would yield:

$$((\,X\,/\,(\,X\,/\,2P\,))\,/P)\;100 = 200 \text{ percent.} \qquad (3\text{-}8)$$

Comparing times used between the two parallel systems, it is seen that the process using multi-algorithmic acceleration takes one half of the amount that was required for the first level parallel process, both using a total of P processors. This capability may exist for the entire problem, or may be obtained in only one portion of the algorithm as the solution progresses to completion. The second operation, discussed in Chapter V, holds this kind of iterative leverage multi-processor improvement.

Building a general mathematical model to predict the outcome of a solution using multi-algorithmic acceleration is more difficult than first level parallel modeling. As can be seen from the above scenario, the procedure is sensitive to the problem being solved, and the level of cooperative interaction possible.

It could be an iterative algorithm beginning with a set of orthogonal vector guesses, exchanging convergence information. The best initial guess finishing first, and those behind but converging to the same point would be stopped by communication of interim data. It could also be a set of different algorithms that solve a problem from greatly different points of view. The greatest speedup obtained being a function of the data involved, the starting procedures, the optimum match of one particular algorithm to the data, the interaction and quality of the interim information, etc.

The number of processors assigned to the different computational efforts, and those assigned to the different algorithms, would be determined statistically or dynamically depending on the parameters of the given problem, the state of progress toward the solution, and the efficiency and speedup attainable at the first level of parallelization.

Stability and accuracy can also conceivably be improved, because yet other available processors can be assigned to controlling the converging processes and refining preliminary answers using substitution techniques along with

searching the maximum and minimum bounds. This would add yet another dimension to multi-algorithmic acceleration procedure.

The objective here is not to only find the quickest serial and perhaps the most accurate solution in accordance with established parameters. Granted, because of the dynamic interaction of all of the cooperating processors, there is a greatly improved probability of it being discovered. But further, a unique solution that has an a priori undetermined path, can accelerate convergence beyond traditional efficiency measures.

This kind of software package demands a MIMD machine because of the requirement to split the computer into different parallel systems going into different directions, but with cooperation between the multiple algorithms. This approach is relatively complex and must be specifically adapted to each particular problem. This does not rule out the generality of the multi-algorithmic procedure. In fact, multi-algorithmic capabilities, non-cooperating and cooperating, may be close to the functioning of future designs of parallel processing machines (Ipsen, 1985).

This approach was included in the parallelization of the new DOA algorithm. The eigenanalysis of the sample covariance matrix shows an extremely valuable multi-algorithmic acceleration speedup activity. It will further be seen that a unique parallel approach at estimating the number of incoming waves has been obtained through a multi-algorithmic application. Since this is the material of Chapter V, it will be put aside for a time to continue on with the last section of this chapter.

Mathematical Model

The method followed in each task was to first analyze the problem and apply

the theory to resolve the problem in the fastest possible way. Mathematical models were constructed which predict the amount of parallel speedup possible, given the problem constraints. This requires a serial model, then a parallel model to be formulated. Finally, given reasonable performance was predicted, parallel code was developed to implement the theoretical models to show by example what was developed in theory.

The experimental data obtained in this work has shown that a reasonably accurate guidelines can be made by analyzing the algorithms by counting multiplications, summations, and comparisons and treating each of them as if they take about the same average time. The total number will be used to mathematically model the number of floating point operations necessary to solve the problem. Additional values that represent the message transfers and excessive serial code necessary will be included into the estimate for the overall computer time used. This procedure is valuable in comparing theoretical speedup ratios and speedup differentials in search of the optimum number of processors to apply. It gives an early indication if parallelization is a reasonable objective for the problem at hand. No attempt is made to include the additional number of instructions necessary to actually implement the algorithm. It is assumed that no additional order increase will result during the actual implementation.

For a large computational problem, the total number of computations is directly related to the overall computer time used. As in the earlier modeling, it will be assumed that the processors are all equal in their computing speed. However, additional parameters will be included for time used in message transit and any strictly serial coded portions if the order exceeds N.

To have realistic speedup differential values for algorithmic decisions, the model will be using 5.7 microseconds as the time for an average single floating

point operation. This implies a single processor capacity of approximately 0.175 megaflops for single precision real data type computations. This is faster than the time used in the earlier examples, but it is a result of the experimental data achieved with the research parallel processor and implemented algorithms. When 16 processors are applied this yields a total of a 3 megaflop computing capacity. Note that this is well below stated clocking speeds, or the typically advertised computing rates because it is an actual average performance value of these specific algorithms. To truly benchmark this performance against another computer, these same programs would need to be transported to it and the run times compared.

Therefore, this flop time is only made available to make relative conclusions in this study and is not to be considered an accurate representation of absolute times outside of the model assumptions. This time will obviously vary between different computers, and even between different computations on the same computer. This data is provided to yield a reasonable value to deal with during this modeling process. Since each operation has actually been implemented, the actual resulting time values will be included as performance tables and can be found at the end of each chapter.

A numerical value to represent the average communications delay for a minimum length message, μ, has been estimated at 50 flops. This value for μ resulted from actual experiments that were run upon the IPSC/2 parallel computer. It is an average that includes multi-hop and single hop communication requirements and message setup time. This indicates that it takes 50 times more computer time to send a message than to compute one floating point operation. Since message length is also a factor, a multiplier coefficient of $\log_2$(length) will be used when length in computer words is known to vary greatly above the minimum value. Using data from the Intel Corporation

and timing experiments on the research computer, this value is a reasonable approximation (Intel, 1987).

Finally, most of the values used will be complex numbers. Rather than dealing with FORTRAN complex math types, the well known relationship between the two by two matrix of complex numbers, and the four by four matrix of real numbers will be applied (Parlett, 1980).

Hence, a two by two Hermitian matrix can be used directly in terms of real values for computation as follows:

$$\begin{bmatrix} (a1 + jb1) & (a2 + jb2) \\ (a2 - jb2) & (a3 + jb3) \end{bmatrix} \quad , \quad \text{where b1 and b3 =0.0 ,}$$

can be written as:

$$\begin{bmatrix} a1 & -b1 & a2 & -b2 \\ b1 & a1 & b2 & a2 \\ a2 & b2 & a3 & -b3 \\ -b2 & a2 & b3 & a3 \end{bmatrix} \quad , \quad \text{where b1 and b3 =0.0.}$$

This means that for a dimension X of the problem when complex numbers are involved, a 2X dimensioned real data matrix will be actually be used. The impact of this is that over four times the number of computations are necessary compared to a strictly real number case.

This choice of complex representation was made because it has the effect of improving the first level parallelization effort by improving the ratio of calculation to communication while improving the speed and computer portability of the final procedures. It also causes true symmetric matrices to occur instead of Hermitian matrices as seen above and therefore requires less direct computations by the algorithms taking advantage of the element symmetry. Using the FORTRAN complex number types does not allow this advantage, because the compiler always generates at least four floating point multiplies

and two floating point adds, and uses two temporary variables for each complex multiply completed (Intel, 1987).

Each of the four tasks outlined in Chapter II take a varying amount of time of the overall algorithm. As the task times are decreased due to parallelization, it should not be forgotten that the improvement of one portion of the overall algorithm by a speedup ratio of Sρ, does not improve the overall algorithm by a speedup ratio of Sρ. To see this, if the paralleled serial portion originally took one fourth of the total serial time, then the improvement of a 128 speedup ratio for that section would yield a overall speedup ratio approximately 1.33

$$S\rho = 100/((25/128)+75) \cong 1.33 \qquad (3\text{-}9)$$

If instead only a speedup ratio of 16 was achieved in that same section, then the overall speedup ratio improvement would be approximately 1.31.

$$S\rho = 100/((25/16)+75) \cong 1.31 \qquad (3\text{-}10)$$

To reach the first level speedup ratio of 128 will take at least 128 processors, and as was seen earlier, it will always be more because almost all problems require some finite amount of serial processing and some communication between nodes.

On the other hand, reaching a speedup ratio of 16, only requires 16 or more processors, which would leave almost 112 of the processors for other tasks. Hence, this could be the better option depending on the speedup differential and efficiency obtained versus other workload possibilities. This fact can have significant bearing on how many parallel processors are used at any particular task at any particular time in the solution.

Granted, in an SIMD computer organization, no other algorithmic tasks can be performed simultaneously, so this is not a design issue. In the MIMD organization however, this is another factor that explains where processors for a multi-algorithmic solution can be expected to come from without causing a

significant degradation in first level speedup while obtaining multi-algorithmic acceleration.

In the following chapters, speedup ratio and computer time used tables will be provided for each task as the task is analyzed and modeled. As previously stated, the actual implementation performance data will be provided rather than the model numbers. Estimates and simulations will be based on 32, 64, and 160 samples per antenna except when the sample size is not a variable. All data will be run over a range of 1,2,4,8, and 16 processors. Data for N equal to 16, 64, 96, and 160 antenna elements will be provided for each task, with the separation between antenna elements one half of a wavelength of the arriving wave unless stated otherwise. Sixteen antennas will be representing small sized antenna arrays, and the other three will represent medium to large antenna arrays. Whenever it is appropriate, multiple signal arrivals will be included to analyze their effects on the speedup parameters.

It should be understood that considerable research energy is expended in reducing the task's computational order, and to then parallelize the resulting algorithm. The implementation of the FORTRAN code is not to be considered the best or the only way to implement the algorithms, nor even the optimum computer language to implement the process. This portion of the research effort was provided to present a demonstrable output of real world estimation performance and bear out the theory derived.

The procedures were timed over a wide range of antenna sizes, 16 to 160, and allowed a similar range of samples per antenna, 32 to 160. Hence, this implementation can be considered to be a compromise design for variable input data rather than tailored to one particular system description. Although it may not be optimal, the implementation is accurate and true numerical representation of the capabilties attained by applying the theory and

developments of this research. Further, It is at an order level that is representative of the dramatic improvement possible. Any additional time saved due to simply enhancing the code, will be significantly less than the improvements already gained by creating the new procedures and the parallelization efforts of this dissertation.

With these parameters, assumptions, and values established, Chapter IV will begin the process of parallel analysis and DOA algorithm design. Task 2, that of computing the estimate of the sample covariance matrix will be the first design topic.

CHAPTER IV

SPATIAL SAMPLE COVARIANCE MATRIX

Following the order of the four operations of the MUSIC solution provided earlier in Chapter II, the first task to be evaluated for parallel speedup is the computation of the estimate of the expected value of the spatial covariance matrix, previously labeled the sample covariance matrix. It is computed from the sample data vectors, $\underline{x}_S = [x_S(1)x_S(2) \bullet \bullet \bullet x_S(N)]^T$, where s = 1, 2, . . . , S, taken from the N antennas in accordance with Equation (1-10).

Parallelization of this task includes some parallel solution characteristics similar to the summation example problem that was presented in Chapter III. The expansion of the requirement to compute, transport, and sum the N by N matrices instead of integers has proved to be quite measurable.

Serial Sample Covariance Matrix Computation

Taking the serial case first, with N antennas, there are N^2 elements that must be computed for each N by N matrix term, from each of the S sample vectors. Because each element value is a complex number, the calculation requires two multiplies and one sum to compute each element of the S different matrices. Further, and still because the matrices are complex, the matrix's dimension represented in the computer is actually 2N by 2N. Hence, in actual application there are $4N^2$ elements per matrix, which requires a total of

$$12SN^2 \text{ flops} \tag{4-1}$$

to compute every element for each of the S sample covariance matrices before it is possible to average them. This is a very large number of computations as N and S exceed 150, and it turns out that the order of this operation is seen to be twelve times an N-cubed order. Improvement in reducing the number of N-cubed computations of this task will show a dramatic improvement in the time used in this operation and overall.

The first reduction can be obtained by observing that the actual sample covariance matrix is Hermitian, but because of the choice of representation of the complex numbers using real elements in the expanded form, the matrix has a true symmetric representation instead of having complex conjugate symmetry. Hence, taking the matrix symmetry into account, only $2N^2$ plus N element values actually need to be computed.

Further, real number representation of the complex elements also causes half of the matrix elements computed to be copies (or complemented copies) so their values do not need to be computed, but can be equated into position. This lowers the total number of elements needed to be directly computed to N^2+N. This yields a total computational savings of $3N^2-N$ flops for each and every sample matrix computed. Figure 11 illustrates the savings associated with using the symmetries for a single matrix with an N equal to 4.

It is obvious that there is an additional cost in computer time required to transfer the computed data into their proper positions, but these transfers will be a function of the faster computer instruction time, not the floating point operation time. Much more significant is that there will only need to be $3N^2-N$ total transfers necessary. This is because these transfers can be completed after all of the computations and summations of all of the sample matrices are completed. Hence, the number of transfers are not a function of the number of samples, S. Normally, S is very large compared to N, and often exceeds it

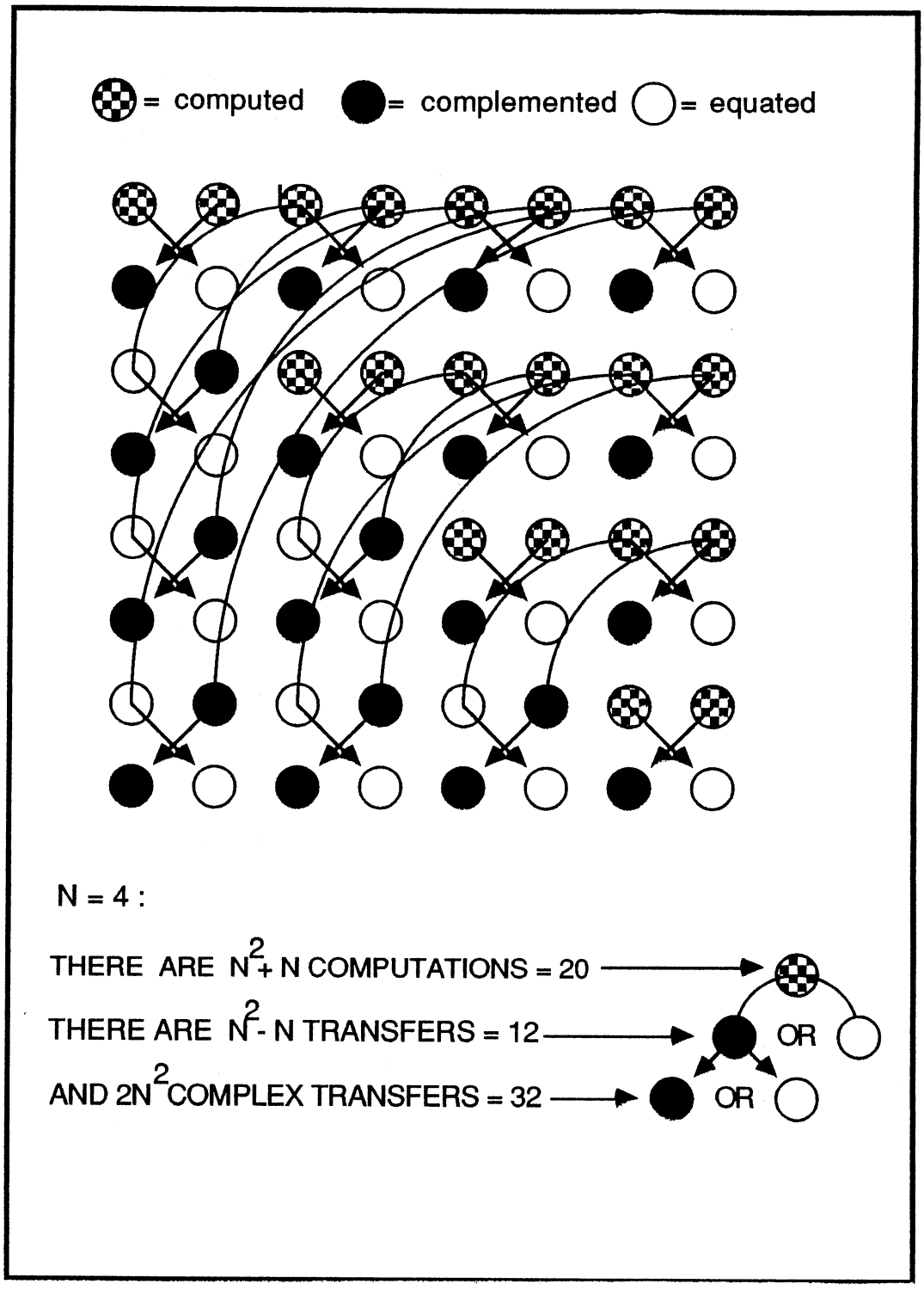


Figure 11. Matrix and Complex Term Symmetry

greatly, so there can be a large time savings expected in this computational adjustment. This causes no trade off in accuracy or precision, hence this is a highly desirable improved serial method.

Taking advantage of the symmetry that exists, the computation of all of the S matrices equals:

$$3S(N^2+N)+\sigma(3N^2-N) \quad \text{flops.} \tag{4-2}$$

where σ represents the ratio of instruction speed to a single floating point operation. Each of these matrices must be added, element by element. This would require an additional

$$(S-1)(N^2+N) \quad \text{flops.} \tag{4-3}$$

The averaging process has the final 2N by 2N matrix divided by the number of samples, S, which would require N^2+N flops because it would be done just before the symmetric values are copied into place. However, the next task is to locate the eigenvectors as was defined in Chapter II. Since eigenvectors and eigenvalues are only found to scalar multiples, it is not necessary to complete the (N^2+N) divides in order to compute the eigenvectors (Parlett, 1980). This will simply result in each eigenvalue being multiplied by S. The eigenvalues could be adjusted with N divisions if the actual eigenvalues are required.

Further, not dividing by S during the process is seen to provide a slight computational improvement over completing the divisions. This is because even though the values are always subject to round off to the computer register length employed, no additional irrational numbers need to be approximated at this point as the result of a division by the number of samples, S, that is other than a power of two.

The final total number of flops required to serially compute the sample covariance matrix from the S samples for N antennas is:

$$(4S-1)(N^2+N)+\sigma(3N^2+N) \quad \text{flops.} \tag{4-4a}$$

For large S and N, which is the target of this research, Equation (4-4a) can be approximated as,

$$4SN^2 \quad \text{flops,} \qquad (4\text{-}4b)$$

which will be used in the speedup ratio and speedup differential parameters to be modeled later. With this serial task optimized for symmetry, the next step is to resolve this same task for parallel computation.

Parallel Sample Covariance Matrix Computation

The first observation for parallel computation of this task is that the operations are computationally independent between samples. Further, the number of samples is usually high, especially in the low SNR case, so S is at least greater than the number of processors. Generally, the lower the SNR expected, the larger the S that is needed. Hence the multi-splitting of this operation for the parallel algorithm could be selected between the different S samples for each node.

One weakness with this approach is that it requires the partial matrix sums completed at each node to be communicated between nodes for completion of the matrix summation. When summed at a single node, this communication takes an equivalence of an additional $\mu\log_2(N^2)\log_2(P)$ flops of time delay, where μ corresponds to the time for a single minimum size message. Then a distribution of the final multi-split matrix to each of the nodes will be necessary in preparation for the next task. This communication can be modeled as $\mu\log_2(N^2/P)\log_2(P)$ additional flops.

Figure 12 illustrates the computations and data flow used in this parallel approach. Notice that the matrix and complex symmetry can be taken advantage of with this parallel approach, however it requires a strictly serial

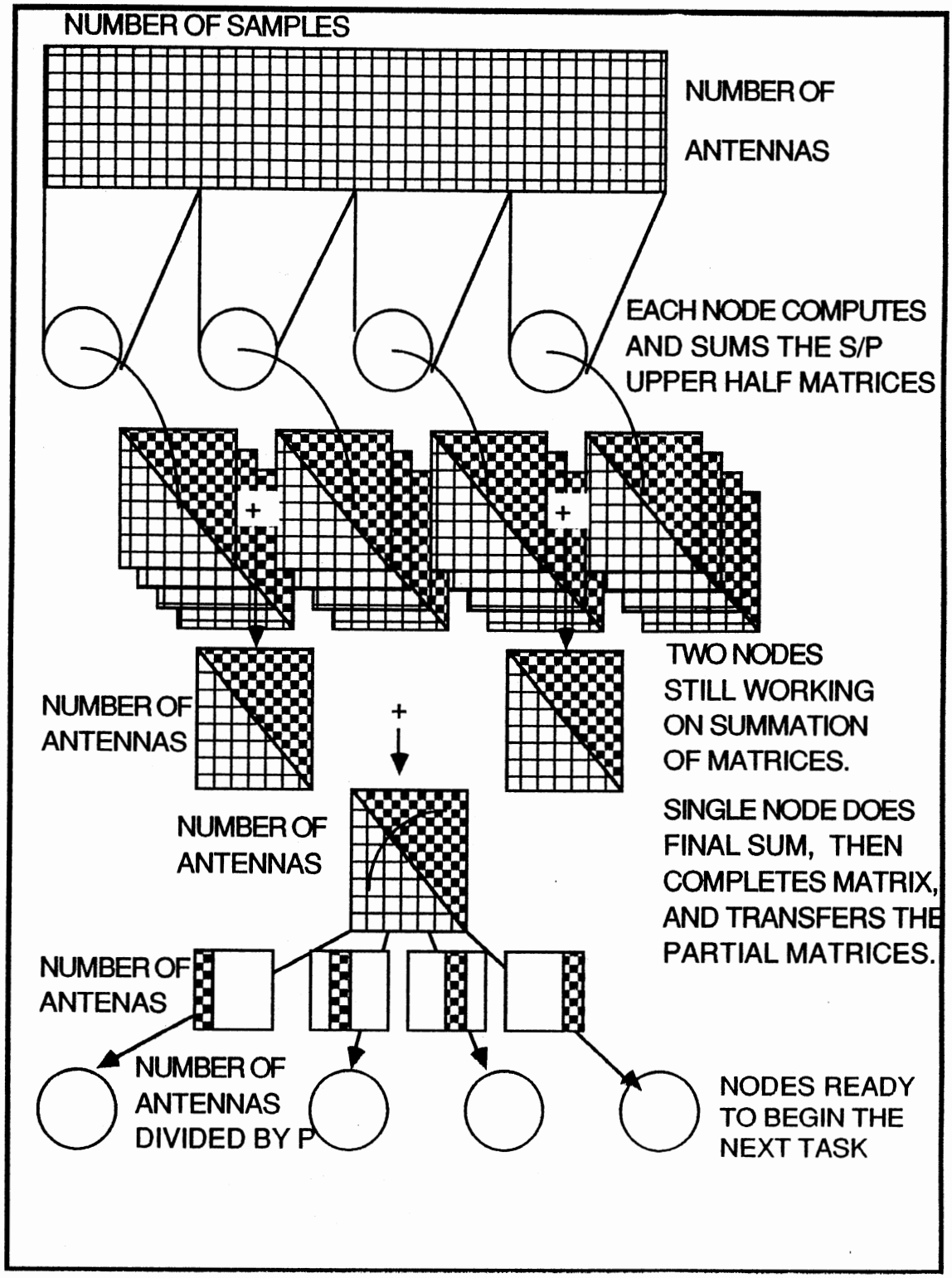


Figure 12. Parallel Split by Samples, Data and Message Flow

activity of transferring the matrix values in place before the split matrix is distributed to the nodes.

A second observation shows that it would also be possible to split the operation by computing the partial multi-split sample covariance matrix at each node. Each node would need all the samples to do this, but no additional communication of the matrix would be necessary to do the final summing, and no distribution communication of the partial matrices from a central node would be necessary after the matrix is computed. This would be an independent "in place" operation and has potential to provide savings over the communications required in the first procedure suggested. This situation is sometimes called a *perfectly parallel* approach indicating total computational independence and no communication requirements, and it can show 100 percent efficiency with speedup ratio numbers equal to the number of processors applied. Figure 13 is a block diagram that illustrates the data flow and computations when following the second approach.

The disadvantage with the second approach is that even though it shows a savings in communication over the first procedure, it can not take advantage of the symmetry of the matrix seen earlier in Figure 11. This causes $3S(N^2-N)/P$ additional concurrent computations to be made compared to the first method.

Figure 14 is a computed graph for variable values of N and P and a fixed S equal to 40 samples per antenna. The two family of curves are comparing the extra computations required in the second procedure to the time consumed in mesages and the strictly serial additions required in the first procedure.

It can be seen that for very small N, and small S, the second procedure would be preferred due to its smaller computational burden increase compared to saving communication time. In fact, when the samples are less then P, or even as low as one, as in an adaptive approach that computes the DOA

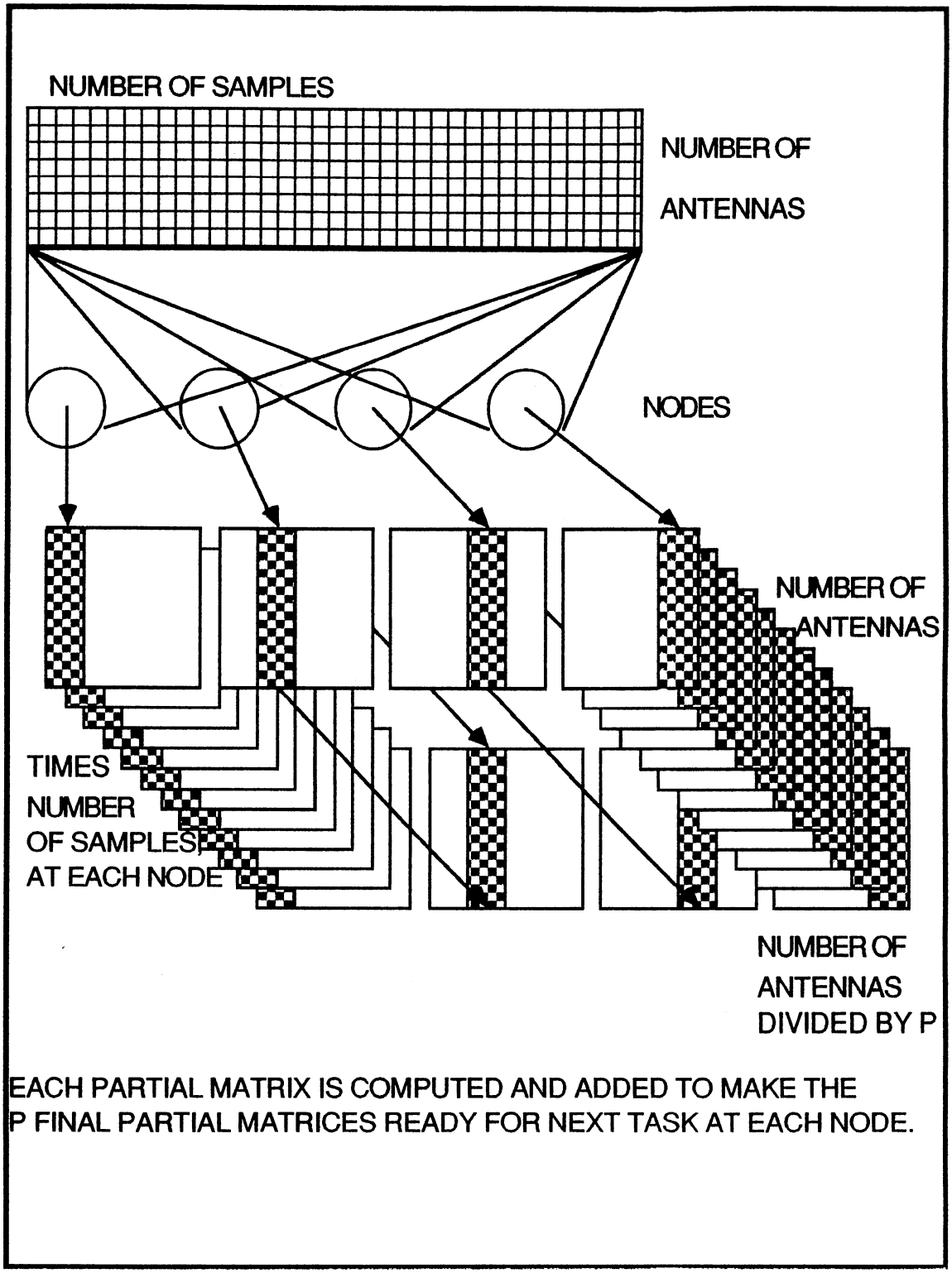


Figure 13. Parallel Split by "In Place" Matrix Computation

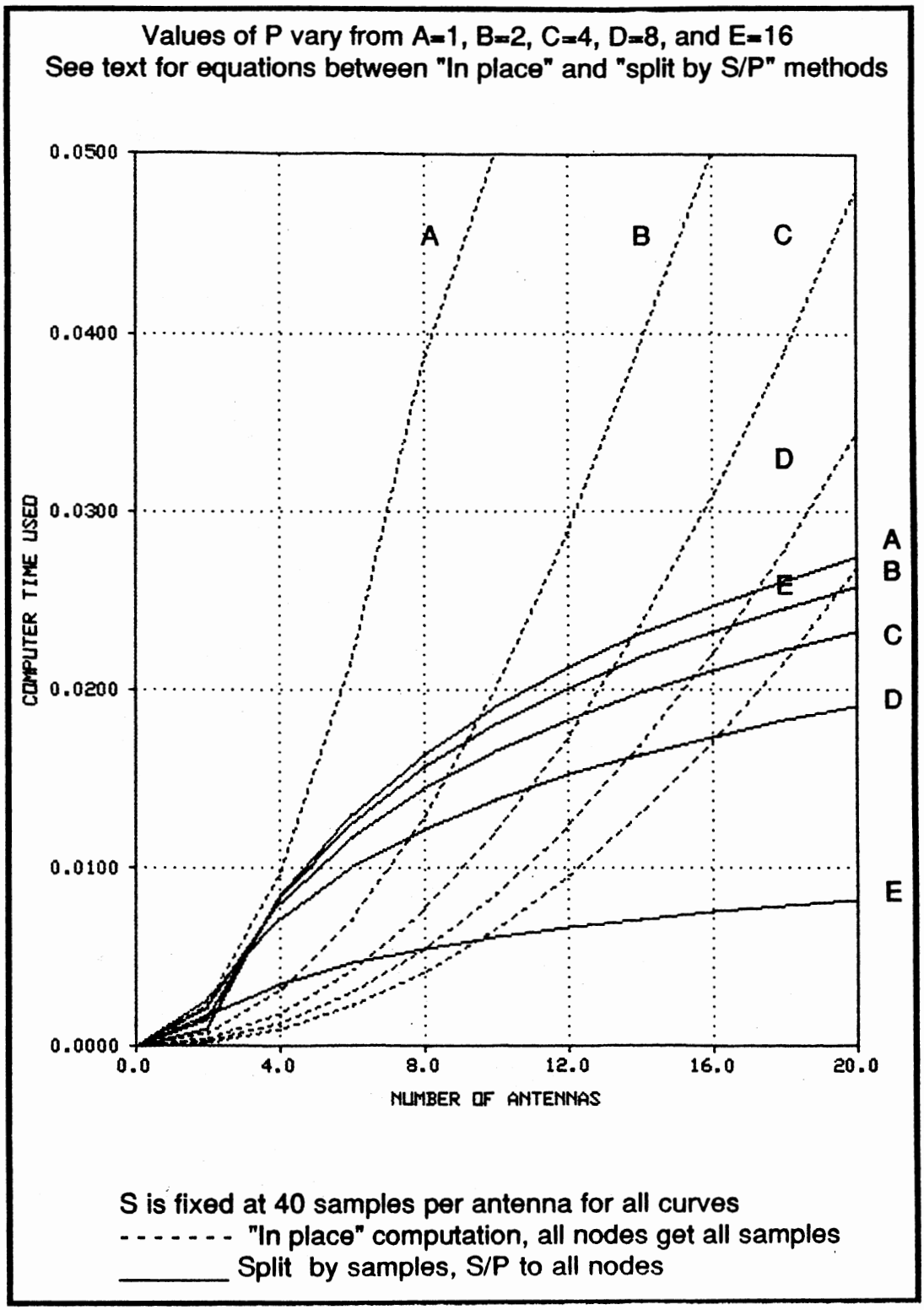


Figure 14. Comparison of Computations Required Between Methods

recursively or adaptively updating with each sample, the second approach would be the required procedure for a parallel implementation (Reddy, 1982, Fuhrman, 1987).

As the values of N, and S increase, the savings of the computations by employing the matrix symmetry will be significantly greater than the time required for message transportation and equating the final numbers into position. Figure 14 shows some cross over points between the two methods.

Another consideration between these two methods is the storage space required in the nodes' memories. The first procedure needs SN/P snapshot samples to be stored at each node, and then a matrix of size $2N^2$ must be held at the supervisory node, at least temporarily. The second method requires all the samples at each node, 2SN elements, but only needs $2N^2/P$ words of storage space for the matrix. It is seen that the memory storage requirement is dependent on the relationship between the S and N values, with neither being seen as a dominant factor when they are approximately equal. As S increases beyond N the first method would once again become the preferred choice, and when S is very small, the second method would be the better choice.

Future task analysis will keep this decision in context, but since the first method is considerably faster for all but very small S and N, it will be considered the preferred choice. Only the first method will be modeled in parallel to be compared to the serial solution. This is the situation referred to in Chapter II where parallel efficiency and speedup ratio parameters are not the most accurate indicators as to the optimum parallel choice. The speedup differential shows that the CTU is lower by following the first choice.

The parallel time in this task needed to compute the S matrix elements using P processors would be

$$(3S/P)(N^2 + N) \quad \text{flops.} \tag{4-5}$$

The partial summing of each of the sample matrices can be accomplished at each node, then as the partial sums are forwarded they require an additional $\log_2 P$, N^2+N sums, and $\log_2 P$ messages each taking $\mu\log_2(N^2)$ amount of time. The distribution of the multi-split matrix for the next task will take $\log_2 P$ $\mu\log_2(N^2/P)$ equivalent flops. This activity will require

$$((S/P)-1)(N^2+N)+ (N^2+N)\log_2(P)+ \mu\log_2(N^2)\log_2(P)+ \mu\log_2(N^2/P)\log_2(P) \quad \text{flops.} \tag{4-6}$$

The averaging divisions are still not required as in the serial case. Leaving a parallel total up to this point as,

$$(3S/P)(N^2+N)+ ((S/P)-1)(N^2+N)+ (N^2+N)\log_2(P)+ \mu\log_2(N^2)\log_2(P)+ \mu\log_2(N^2/P)\log_2(P) \quad \text{flops .} \tag{4-7}$$

The noncomputational strictly serial portion of the code for Task 1 is negligible compared to the paralleled computational tasks, so it is set to zero. However the transfers associated with the symmetry of the matrix is a strictly serial activity that must be completed at the supervisor node, and requires

$$\sigma(N^2-N) \quad \text{flops.} \tag{4-8}$$

The complex symmetry associated with the representation of the complex field can be done in parallel at each node after the multi-split matrix is distributed and is equivalent to

$$\sigma(2N^2/P) \quad \text{flops} \tag{4-9}$$

The total time, in flops, for the parallel process of computing the estimate of the sample covariance matrix, $\mathbf{R}_x$, from the S sample snapshots using the N antennas applying P processors would be:

$$(3S/P)(N^2+N)+ ((S/P)-1)(N^2+N)+ (N^2+N)\log_2(P)+ \mu\log_2(N^2)\log_2(P)+ \mu\log_2(N^2/P)\log_2(P) + \sigma(N^2-N) + \sigma(2N^2/P) \quad \text{flops .} \tag{4-10a}$$

As with equation (4-4b), approximation yields a simpler and yet reasonable value which will provide an accurate picture of this operation with large N and

S. Hence,

$$4SN^2/P + (N^2+100N)\log_2(P) \quad \text{flops} \qquad (4\text{-}10b)$$

will be substituted for Equation (4-10a). The first term represents the parallel speedup improvement with P processors, and the second term shows the processing time degradation due to the message handling and additional serial computations.

Equations (4-11) and (4-12) below model the speedup ratio and speedup differential respectively using Equations (4-4b) and (4-10b) for the computation of the sample covariance matrix:

$$S\rho : \quad \frac{4SN^2}{4SN^2/P + (N^2+100N)\log_2(P)} \quad \text{flops} \qquad (4\text{-}11)$$

$$S\delta : \quad 4SN^2 - 4SN^2/P + (N^2+100N)\log_2(P) \quad \text{flops} \qquad (4\text{-}12)$$

Notice that the approximation and actual equation from the parallel models, equal the approximation and actual equation respectively, from the serial models, if the number of processors, P, equals one.

Sample Covariance Matrix Computation Results

Table 1 is the data showing the CTUs (rather than speedup differential) in seconds and speedup ratios, $S\rho$, resulting from computing the sample covariance matrix for varying values of S, N, and P. Table 1 is found as the last page of this chapter. This placement allows for easier reference to the results which is preceded by the discussion. Each set of results obtained will likewise follow the disscusions and be the last pages of the future chapters when applicable. It can be seen that the first level parallel improvement of task is close to P when P is 2 for all values of N and S used. As N increases, causing

more calculations per message, the parallel efficiency is improved for the higher values of P. For larger S values the data also shows better efficiency and the speedup ratio will more closely approximate the value of P.

The table indicates that this first task of conversion to a parallel algorithm provides reasonable efficiency for values of N above 64. The values of N less than 16 show a much larger amount of wasted computer power.

Providing additional practical information about this task are the CTU values. What is obtained is significant in the decision making process. The serial time is when P equals one, and the various other P values show the speedup improvement (or loss) as a result of increasing the number of processors, P.

Hence, Table 1 reports that the serial process for Task 1 takes at the lowest end, 0.211 seconds using an N value of 16 and S=32, and at the highest end, 94.90 seconds when N=160 with S=160. It is significant to remember that this serial task is performed in a somewhat traditional way, except perhaps taking advantage of the symmetry. Hence, the computer time savings here are associated with the application of the parallel processor to the problem at the first level and not with a major modification of the serial procedure.

The greatest improvement in speedup differential is found in the largest parameter case with 87.95 seconds out of 94.90 seconds of processing time being saved. For an array size of 16 receive elements, the smallest simulated, and 160 samples, the absolute time saved ranges from one half of a second to almost the entire serial time, which is slightly over one second. Of course, once again this CTU data depends on the average flop time for the research computer. A computer having a flop time different than this value will result in a different amount of time that can be saved. However, the general shape of the curve and conclusions will remain valid as long as not too great a change in flop time occurs.

One concludes then, that the first task shows good parallel improvement at the first level of parallelization using the larger values of N, and S, and smaller to medium P values of the range implimented.

There seems to be reasonable evidence that there is parallel computer power available for multi-algorithmic processing with smaller antenna arrays. However, because of the nature of this operation's computation, deterministic multiplications and sums, there is no advanced algorithmic opportunity recognized to accelerate the solution by applying cooperative multi-algorithmic techniques. Therefore, when computing the sample covariance matrix in the parallel mode, maximum available processors should be applied at the first level of parallelization. For N less than 16, and using a low sample size, it is highly probable that the parallelization threshold would be reached and a slowdown situation would occur. This is also possible when applying more than 16 processors to the problem even with larger arrays.

In the next task, the eigenanalysis of the computed matrix, it will be seen that there is parallel algorithmic opportunity at the first level of parallelization, and due to the nonlinear nature of the eigenanalysis solution, there is also parallel improvement opportunity at the cooperative multi-algorithmic level.

TABLE 1

COMPUTATION OF THE SAMPLE COVARIANCE MATRIX

32 SAMPLES FROM EACH ANTENNA

Number of Nodes	Number of Antennas 16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	0.211	1.00	3.23	1.00	7.23	1.00	20.61	1.00
2	0.118	1.79	1.70	1.90	3.79	1.91	10.46	1.97
4	0.072	2.93	0.95	3.40	2.09	3.45	5.71	3.61
8	0.051	4.14	0.58	5.57	1.26	5.74	3.40	6.06
16	0.041	5.15	0.41	7.88	0.87	8.31	2.31	8.92

64 SAMPLES FROM EACH ANTENNA

Number of Nodes	Number of Antennas 16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	0.408	1.00	6.22	1.00	13.94	1.00	38.89	1.00
2	0.216	1.89	3.19	1.95	7.14	1.95	19.74	1.97
4	0.122	3.34	1.69	3.67	3.76	3.70	10.35	3.76
8	0.075	5.44	0.95	6.52	2.01	6.65	5.72	6.80
16	0.054	7.55	0.59	10.47	1.29	10.82	3.47	11.21

160 SAMPLES FROM EACH ANTENNA

Number of Nodes	Number of Antennas 16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	1.005	1.00	15.19	1.00	34.06	1.00	94.90	1.00
2	0.510	1.97	7.69	1.98	17.20	1.98	47.57	1.99
4	0.269	3.74	3.94	3.86	8.79	3.87	24.28	3.91
8	0.149	6.74	2.05	7.32	4.61	7.39	12.68	7.48
16	0.090	11.17	1.15	13.15	2.54	13.36	6.95	13.65

Note: CTU is in seconds, S_ρ is the speedup ratio

CHAPTER V

EIGENSTRUCTURE DECOMPOSITION

It was shown earlier that by using what is equivalent to the average orthogonality of the entire set of noise eigenvectors within the MUSIC peaking function, Equation (2-9), it was possible to compute a discrete amplitude plot containing peaks which represent the bearings, or the DOAs of the incoming wavefronts.

The eigendecomposition of the sample covariance matrix to extract the eigensystem is usually considered a major computational burden in MUSIC because of the N-cubed eigenanalysis procedures typically required.

A significant reduction in this workload is accomplished within this research when using large passive antenna arrays because the eigenanalysis is reduced to finding only the maximum eigenvector and minimum eigenvector. Chapter VI provides insight into the creation of a functional which uses only these two eigenvectors in the estimation process.

This chapter assumes the two eigenvector estimation approach as valid, and is therefore only concerned about how to most rapidly get these two unknown eigenvectors from the sample covariance matrix. For few eigenvectors the most efficient method is an N-squared order iterative algorithm known as the power method. It will be seen that a multi-algorithmic acceleration procedure can be applied to speed convergence process and also provide an estimate of the number of arriving waves.

Power Method

A straight forward step by step description of the actual process of the power method begins with a N by N matrix, **A,** and an arbitrary guess at the maximum eigenvector, $\underline{x}_1$. The guess, or starting vector, can be anything except the trivial case of all zeros, but a reasonable first choice is usually all ones. If a better guess of the maximum eigenvector is known, it will advance the convergence process, but this is not necessary to have convergence for a symmetrix matrix as is found in this case (Conte, 1980).

The first step is to calculate $\mathbf{A}\underline{x}_1$ to yield a vector called $\underline{y}_1$ ($\mathbf{A}\underline{x}_1 = \underline{y}_1$).

The second step is to find $| y_i |_{max}$, the maximum entry in magnitude of the computed vector, $\underline{y}_1$.

The third step is to normalize the vector $\underline{y}_1$ by dividing all the elements by $|y_i|_{max}$. This gives

$$\mathbf{A}\underline{x}_1 = \underline{y}_1 = \eta_2\underline{x}_2 , \tag{5-12}$$

where η_2 is the signed element that was used to normalize $\underline{y}_1$.

The fourth step is to test the new value, η_2 , (or vector, $\underline{x}_2$) and see if it is sufficiently close to the previous value, η_1, (or vector, $\underline{x}_1$). Each iteration brings the new value closer to the maximum eigenvalue (or maximum eigenvector) which can be seen by the convergence to the eigenvalue and its associated eigenvector. The tolerance used in this work is .01 percent. This was used considering the analog to digital conversion of the samples and typical receiver output capabilities. Convergence is completed when the iterates are within .01 percent of each other.

If another iteration is needed the new matrix-vector product,

$$\mathbf{A}\underline{x}_2 = \underline{y}_2 = \eta_3\underline{x}_3, \tag{5-13}$$

is formed to yield $\underline{y}_2$ equal to $\eta_3\underline{x}_3$ where η_3 is the signed element used to

normalize $\underline{y}_2$, and $\underline{x}_3$ is the normalized vector as was done in the last iteration. The process continues until $\eta_i \underline{x}_i$ converges to the dominant eigenset $\lambda_{max} \underline{e}$. The iterations can be stopped when $\underline{x}_i$ of the last iteration is sufficiently close to $\underline{x}_{i+1}$ of the next iteration, or if η_i is sufficiently close to η_{i+1} (Conte, 1980).

The power method is an iterative process and the time required for convergence, is highly dependent on the ratio of the magnitude of the next largest eigenvalue to the magnitude of the largest eigenvalue, and the closeness of the starting vector to the maximum eigenvector. The greater the separation of these two eigenvalues, and the closer the starting vector, the faster the convergence is completed. It is true that convergence can be slow in some cases, but acceleration techniques exist that maintain the N-squared order of the calculation. In any case, convergence is guaranteed for a symmetric matrix (Parlett, 1980).

Later in this chapter the power method implementation will be analyzed again when deriving the mathematical model for the number of calculations that are expected. This will provide a further description of the algorithm which is sufficient for those familiar with the method. For others, there are many texts that provide detailed looks at the power method of eigendecomposition (Wilkinson, 1965, Anton, 1977, Conte, 1980).

A key to using the power method for the parallel procedure can be seen in that the computational routines that make up the working part of the power method are matrix-vector multiplications. These computations are iterative in nature, and have an intrinsic make-up more favorable for the first level of parallelization then many other decomposition methods.

The familiar procedures based on the Jacobi method, for example, are successive sequences of plane rotations in an effort to transform the matrix to diagonal form. The nonparallel nature of this procedure exists because the

requirement to have the results of the last modified matrix elements before the next set of elements can begin computation. This does not allow the matrix to be split and operated on in an effective concurrent manner.

Recent work with parallelization of the Jacobi method has beared this out by showing that even when using specialized one-sided Jacobi procedures for solving eigenproblems on parallel architectures, the time will be near 50 seconds using 16 processors for a real matrix of order 64 (Eberlein, 1987). It can be seen that following this procedure, this one task would then far exceed the real-time and near real-time goals as established earlier.

Generalized Eigenvalue Problem

It was shown in Chapter II that MUSIC requires the solution to the generalized eigenvalue problem between the matrix pair ($\mathbf{R}_x$, $\mathbf{R}_b$) when the noise sample covariance matrix is assumed to be known and not spatially white. It was assumed in the MUSIC development that an estimate of the noise characteristics of $\mathbf{R}_b$ can be obtained a priori for the MUSIC implementation, and it was assumed to have a Gaussian distribution in the MUSIC procedure (Schmidt, 1981). This is not a realistic situation for in virtually all practical applications, such information is never available a priori (Cadzow, 1988). However, reasonable research success has been obtained by using this assumption about knowing the noise characteristics.

Here, the matrix $\mathbf{R}_b^{-1}$ also needs to be estimated a priori, in this case referring to *before calculation* of the generalized eigenset solution. Based on the $\mathbf{R}_b$ estimate already obtained, an estimate of $\mathbf{R}_b^{-1}$ could be made available by completing an inverse and storing the multi-split results in a mapped array awaiting computation. This is in contrast to trying to compute this matrix on-line.

Having an estimate of $\mathbf{R}_b^{-1}$ available is necessary to allow the generalized eigenvalue problem solution to be of order N-squared.

When it is needed to solve between the matrix pair ($\mathbf{R}_x$,$\mathbf{R}_b$), rather than assuming spatially white noise, where $\mathbf{R}_b = \mathbf{I}$, then some modification to the power method needs to be created. This is because the power method is only appropriate for conventional or standard eigenvalue problems.

Remember that $\mathbf{R}_x$ is the result of the first operation in Chapter IV. Also, now instead of assuming only the $\mathbf{R}_b$ matrix is known, this procedure assumes that an estimate $\mathbf{R}_b^{-1}$ is available. Hence, with these estimates at hand, the generalized eigenvalue problem can be changed from ,

$$\mathbf{R}_x\underline{e} = \lambda_{min}\mathbf{R}_b\underline{e} \ , \tag{5-14}$$

to

$$\mathbf{R}_b^{-1}\mathbf{R}_x\underline{e} = \lambda_{min}\underline{e} \ . \tag{5-15}$$

Of course, the matrix-matrix multiplication indicated in Equation (5-15) is still an N-cubed order function. But the computation never needs to be made directly. Instead, the $\mathbf{R}_x\underline{e}$ computation can be completed to yield a new column vector , say $\underline{y}$. Then $\mathbf{R}_b^{-1}\underline{y}$ can be computed. Both of these computations are matrix vector computations, hence one additional N-squared order computation is necessary for each iteration. Again, this procedure can be used to improve the estimate when the maximum eigenvector is being computed using the power method knowing or estimating the matrices $\mathbf{R}_b$ and $\mathbf{R}_b^{-1}$.

A shift of origin is necessary to converge to the minimum eigenset instead of the maximum eigenset when applying the power method. This DOA procedure will eventually need to do both, that is, resolve the maximum and the minimum eigensets. Shifting the origin of the eigenvalue set is accomplished by subtracting an amout equal to the desired shift from each element of the diagonal of the matrix under decomposition. This causes an equal shift in each

eigenvalue, but no change to the associated eigenvector (Wilkinson, 1965). In order to shift the origin when the noise is considered to not be spatially white, but known as it is above, the following procedure needs to be employed that is different from the spatially white case.

Starting with the same generalized eigenvalue problem Equation (5-14), the same first step as above once again results in Equation (5-15). Then an appropriate shift scalar multiplier, τ, is used to compute a new shifted matrix

$$(\mathbf{R}_b^{-1}\mathbf{R}_x - \tau\mathbf{I})\underline{e} = \lambda_{min}\underline{e} \,. \tag{5-16}$$

where $\mathbf{I}$ is the identity matrix of appropriate size. The new eigenvalues of the matrix $\mathbf{R}_b^{-1}\mathbf{R}_x - \tau\mathbf{I}$ will be shifted from the matrix $\mathbf{R}_b^{-1}\mathbf{R}_x$ by an amount equal to $-\tau$, but there will be no change to the associated eigenvector values.

This is the same shifting procedure as would be done in the standard eigenvalue problem, except of course, that the computation of $\mathbf{R}_b^{-1}\mathbf{R}_x$ was never done directly, hence the subtraction cannot be completed directly.

Selection of an appropriate τ to find the minimum eigenvalue is a trivial problem, and will be dealt with in the next section of this chapter. There also exists an optimum shift, τ_{opt}, which causes fastest convergence, but this has no bearing on the method at this point. However, finding τ_{opt} is certainly not a trivial problem. The optimum choice for the shift is not necessary in the present development, and is only introduced here because of its significance later in this chapter. In this case, given τ, Equation (5-16) is premultiplied by $\mathbf{R}_b$ to yield

$$\mathbf{R}_b(\mathbf{R}_b^{-1}\mathbf{R}_x - \tau\mathbf{I})\underline{e} = \lambda_{min}\mathbf{R}_b\underline{e} \,. \tag{5-17}$$

But that equals,

$$(\mathbf{R}_b\mathbf{R}_b^{-1}\mathbf{R}_x - \tau\mathbf{R}_b\mathbf{I})\underline{e} = \lambda_{min}\mathbf{R}_b\underline{e} \,, \tag{5-18}$$

which, in turn, yields

$$(\mathbf{R}_x - \tau\mathbf{R}_b)\underline{e} = \lambda_{min}\mathbf{R}_b\underline{e} \,. \tag{5-19}$$

Therefore, in order to cause the appropriate origin shift to locate the minimum eigenvector, or noise direction vector, when the sample noise covariance matrix is colored, the first step is to multiply the scalar shift, τ, by the sample noise covariance matrix. This matrix, $\tau\mathbf{R}_b$, is then subtracted from the sample covariance matrix, $\mathbf{R}_x$. Both of these operations are N-squared order computations. This resulting matrix then replaces $\mathbf{R}_x$ in Equation (5-14), and the solution again progresses as described above in the original generalized eigenvalue problem.

Optimum Scalar Shift

The discussions above indicated that a scalar shift value, τ, exists that causes the minimum eigenvalue to become the maximum eigenvalue so that the power method could extract the minimum eigenset. It will also be seen that not only does a shift of the matrix to get the minimum eigenvalue exist, but also there is an optimum shift, τ_{optmin}, that increases convergence rate to the minimum eigenvalue solution. Likewise, there is an optimum multiplier scalar shift value, τ_{optmax}, that causes greater speed of convergence when converging to the maximum direction vector.

The general result for the existence of an optimum shift for a power method and the corresponding improvement in convergence is well known, and is defined for the set of eigenvalues associated with the eigensystem of a matrix (Gourlay, 1973). Given a system of eigenvalues that satisfy the relationship, $|\lambda_1|>|\lambda_2|>\ldots>|\lambda_M|>|\lambda_{M+1}|>\ldots>|\lambda_{N-1}|>|\lambda_N|$, depending on the choice of τ, the dominant eigenvalue will be $\lambda_1-\tau$, or $\lambda_N-\tau$. Given the above order, when τ is equal to 0.0 then λ_1 will be the absolute maximum eigenvalue, and the power method will converge to λ_1 and its associated eigenvector. If the value of τ is

chosen as λ_1, then $|\lambda_N - \lambda_1|$ will become the absolute maximum eigenvalue and the power method applied to this shifted matrix will converge to $|\lambda_N - \lambda_1|$ and the eigenvector associated with λ_N. It should be apparent that, in the first case, the maximum signal eigenvector is found, and in the second case the minimum noise eigenvector is located.

The rate of convergence will depend on the ratio $|(\lambda_2)/(\lambda_1)|$ for the maximum and $|(\lambda_{N-1})/(\lambda_N)|$ when searching for the minimum (Gourlay, 1973).

The smaller the ratio the faster the convergence, the closer the ratio is to one, the slower the convergence. When a shift, τ, is applied as is discussed above for the generalized eigenvalue problem or for the normal eigenvalue problem, the rate of convergence of the power method will then depend on ratio of the shifted eigenvalues, $|(\lambda_2-\tau)/(\lambda_1-\tau)|$ and $|(\lambda_{N-1}-\tau)/(\lambda_N-\tau)|$.

For two suitable values of τ, called τ_{optmax} and τ_{optmin}, each of the ratios can be minimized thereby causing the fastest convergence possible for the particular matrix and eigensets. The correct value of shift in each case can be determined from the following two equations. The first is to determine the shift for the maximum direction vector and the second provides the minimum direction vector (Wilkinson, 1968):

$$\tau_{optmax} = (1/2)\ (\lambda_2 + \lambda_N) \qquad (5\text{-}20)$$

$$\tau_{optmin} = (1/2)\ (\lambda_1 + \lambda_{N-1}) \qquad (5\text{-}21)$$

Of course, the problem now is to determine the values of λ_2 , λ_1, λ_N ,and λ_{N-1} so as to provide the τ_{opt} before the iterations begin, not after the eigenanalysis is complete. Hence a method to make an estimate of τ_{optmax} and τ_{optmin} needs to be determined that does not require completion of the actual eigenanalysis itself.

It should be very clear that there is no deterministic procedure which can be applied to solve this problem in general. However, there are some special

cases which occur, and these are extremely important because the values of the optimum shifts for these can be estimated very accurately with very little computation thereby causing rapid convergence. Further, there is a method that showed some success in this research for the rest of the possibilities, and it has been incorporated into the parallel algorithm as a multi-algorithmic accelerated procedure.

Fortunately, the minimum optimum shift value was found to be related to the actual number of arriving waves, M, which is another unknown at this point that is needed before the solution to the DOA problem is complete. Remember that the MUSIC estimate of the number of incoming wavefronts required the extraction of the entire set of eigenvalues to determine M. Of course, all of the eigenvalues are not available in this approach so a new procedure is needed. The procedure developed will become clearer as this chapter progresses and is part of the contribution of this work.

In search of the optimum shifts, it should first be noted that the matrix $\mathbf{R}_X$ was developed from a finite set of samples in accordance with Equation (2-2), hence it must be a positive semi-definite or a positive definite matrix. This indicates that its eigenvalues are nonnegative. Hence the following order of the eigenvalues must be true: $\lambda_1>\lambda_2>\ldots>\lambda_M>\lambda_{M+1}\geq\ldots\geq\lambda_{N-1}\geq\lambda_N\geq 0$. Note that the smaller N-M eigenvalues may be zero, or may equal each other.

Also, the sum of the eigenvalues of a matrix is equal to the sum of the diagonal elements, that is, $\text{trace}=\lambda_1+\lambda_2+\ldots+\lambda_M+\lambda_{M+1}+\ldots+\lambda_{N-1}+\lambda_N$.

Therefore, it can be concluded that each individual eigenvalue must be less than or equal to the trace. Computation of the trace, the sum of the diagonals of the sample covariance matrix, is a simple N order function and can be completed in negligible time compared to the higher order functions.

Given the above, it will be shown that the trace of the sample covariance

matrix can be used to develop reasonable approximations for both τ_{optmax} and τ_{optmin}. There are separate cases to consider for each situation depending on the number of arriving wavefronts.

Maximum Eigenvalue Optimum Shift

When only a single arriving wave is present, then this is the special case of approximating a rank one sample covariance matrix. In this situation the maximum eigenvalue is quickly converged to because the single signal eigenvalue is widely separated from the many, N-M, smaller noise eigenvalues. In fact, in this case the power method will always converge in just two iterations when the SNR is high because the noise eigenvalues will be close to zero.

To see this, consider the origin of the matrix $\mathbf{R}_x = \underline{x}\,\underline{x}^H$. The first step in the power method is to take any vector, $\underline{y}_1$, that is not equal to zero, and compute $\underline{y}_2=\mathbf{R}_x\,\underline{y}_1= \underline{x}\,(\underline{x}^H\underline{y}_1)$. If $\underline{y}_2=\underline{0}$, then $\underline{y}_1$ is an eigenvector belonging to the eigenvalue 0. Otherwise $\underline{y}_2$ is an eigenvector belonging to eigenvalue $(\underline{x}^H\underline{x})$. This is because, $\mathbf{R}_x\,\underline{y}_2 = \mathbf{R}_x\ \underline{x}\,(\underline{x}^H\underline{y}_1) = \underline{x}\,(\underline{x}^H\,\underline{x}\,)(\underline{x}^H\underline{y}_1) = \underline{x}\,(\underline{x}^H\underline{y}_1)\,(\underline{x}^H\,\underline{x}\,)$, hence,

$$\mathbf{R}_x\,\underline{y}_2 = \underline{y}_2\,(\underline{x}^H\underline{x}). \qquad (5\text{-}22)$$

In actual implementation, the algorithm can not finish yet having only the initial eigenvector guess to compare against. So the second iteration is needed to be completed which computes $\underline{y}_3=\mathbf{R}_x\,\underline{y}_2$. Now, comparing the normalized values of $\underline{y}_2$ and $\underline{y}_3$, it will be discovered that $(\underline{x}^H\underline{x})$ is the eigenvalue and the values of $\underline{y}_2$ and $\underline{y}_3$ are essentially within the tolerance of the precision used. Hence, for the maximum eigenvalue with one arriving wave, $\tau_{optmax} = 0.0$, or no shift is the optimum shift.

In the second case, with two arriving waves, it is seen from Equation (5-20) that the optimum shift will be half of the sum of the smaller signal eigenvalue,

λ_2, and the smallest noise eigenvalue, λ_N. For large SNR cases λ_N is close to zero relative to λ_2. With the trace available, being the sum of both signal eigenvalues plus all of the noise eigenvalues, an estimate of the appropriate shift can be derived. It is necessary to know the distribution of the eigenvalues to get the exact optimum shift. If a choice of the distribution is estimated reasonably well and a shift made on that choice, then a faster convergence rate can be obtained, if not the fastest.

An ad hoc distribution choice based on simulation experiments with uniform colinear array situations with two arriving wavefronts is that the dominant eigenvalue is approximately twice size of the other signal eigenvlaue. Using this distribution yields a maximum optimum shift when locating the maximum eigenvalue that is approximately one sixth of the trace. This will be close to the maximum eigenvector optimum shift defined by Equation (5-20).

In the third case, three or more arriving waves, the same distribution assumption is extended by using an equally spaced linear distribution of the eigenvalues to yield the estimated locations of each of the next smaller eigenvalues. This is nothing more than an arithmetic progression with the common difference being M^{-1} times maximum eigenvalue. The values of the same two largest eigenvalues are needed as in the case above, with the situation now being that the trace value is increased by the replacement of noise eigenvalues with the larger, but decreasing in magnitude, signal eigenvalues. The optimum shift for these cases is a smaller and smaller portion of the trace as the number of signals increase.

Based on simulations, it appears that approximately one tenth of the trace could be selected as the shift amount in most cases and convergence would still be rapid. This is due to the large separation of the largest noise eigenvalue and the signal eigenvalue in the first case is normally very fast, hence they

would not be affected much by the shift. In the poor SNR circumstance, convergence could even be faster than without the shift, because the noise eigenvalue will be nearly as large as the smallest signal eigenvalues.

In actual practice, it was found that there was not a dramatic improvement in the number of iterations saved. That is, the loss asssociated with extra iterations in the low noise cases, the more predominant situation, was worse than the improvement obtained in the higher noise cases. Also a consistent correlation between the different number of arriving waves and the correct shift was not established. The differences between the convergence improvements associated with the different possible shifts were too slight to be a reliable indicators as to the number of arriving wavefronts or even a reasonable approach to accelerate convergence.

Hence, for the particular nature of the sample covariance matrix, and the assumed eigenvalue distribution, it was most advantages in all multiple arrival cases to not shift the matrix before iterations begin.

Minimum Eigenvalue Optimum Shift

Contrary to the maximum eigenvalue situation, the optimum shift for the minimum eigenvalue case is more dependent on the input data. It has clearer separation between situations, and it also provides very accurate information about the number of arriving waves. Three cases are considered again.

With a single arriving wave the optimum shift is simply the maximum eigenvalue. This could be a result of the maximum eigenvalue search above, however it is a faster scheme to approximate the shift and begin simultaneous iterations on another subset of nodes of the parallel computer.

The trace, already available, will be close to the maximum eigenvalue with a

single incoming wave, differing only by the sum of the noise eigenvalues. The noise eigenvalues are nearly equal and close to zero relative to the signal eigenvalue except in the poor SNR situations. Hence, if iterations for the smallest eigenvalue begins with shift equal to the amount of the trace, and a single arriving wave is present, then the convergence will be rapid, two to three iterations, and the noise direction vector will be resolved. Notice with this shift, in the single arrival situation, the shifted matrix will still approximate a rank one matrix as discussed earlier. This would not be as clear if the noise eigenvalues are widely separated in value. The shifted matrix would have several different eigenvalues and would have a higher effective rank. This occurs in the very poor SNR case, and causes some difficulty in this procedure.

When there is more than one arriving wave, then the trace value is larger than the optimum shift causing the iteration convergence rate to be at a suboptimal level which leads to the second case.

In the second case, i.e., for two arriving waves, Equation (5-21) shows that the sum of the two signal eigenvalues divided by two would be the optimum shift. The trace is exactly equal to the sum of the two signal eigenvalues plus all of the noise eigenvalues. When the noise eigenvalues are close to zero a very good estimate for the optimum shift for the minimum eigenvalue with two arriving waves could then be obtained by simply dividing the trace by two. Empirical results showed that rapid convergence occurs with this shift amount when two arriving wavefronts are present.

What can be seen, as it is developing, is that the divisor for the shift amount that yields the fastest convergence is related to the number of arriving waves. In the first two cases, the minimum eigenvalue optimum shifts are very accurate indicators and require no assumption about the eigenvalue distribution. The last case, three or more arriving waves, has very good experimental results with

simulations, but is not as accurate an indicator as the first two cases were because an eigenvalue distribution once again needs to be assumed.

To estimate τ_{optmin} for the third case, i.e., for three or more arriving waves, it is required to be able to reasonably approximate the distribution of the signal eigenvalues. Some work has been completed in determining the exact eigenvalue distribution for incoherent Gaussian covariance matrices. As this work has not resolved the issue, heuristic considerations were considered. Arithmetic and geometric progression methods were compared with the arithmetic progression being favored for a uniform spacing (Martin, 1988).

In agreement, the work accomplished in this research favors the ad hoc choice chosen in the maximum eigenvalue search of an equally distributed linear set, or an arithmetic progression with a common difference of M^{-1} times the maximum eigenvalue. This has proven to be accurate in experimental data derived from a uniform colinear array. Following this guideline, the optimum shift for three or more arriving waves can be determined from Figure 15.

Here, as earlier, the divisor for the trace is the estimate to the number of arriving wavefronts. This same table would apply in this work if the number of arriving waves are know a priori. In this case only the required shift would be accomplished and rapid convergence would be achieved.

Experiments have shown that the more arriving waves and poorer the SNR, the less accurate the estimate becomes. Of course this is also true for the methods that attempt to determine the number of arriving waves by analysing the set of extracted eigenvalues as discussed in Chapter II. Because the method is based on a particular distribution, it does limit the algorithm to colinear arrays when the number of arriving waves is unknown. However, it may be possible to achieve similar results for any given array structure by operating on subsets of the array that are colinear. There are several factors

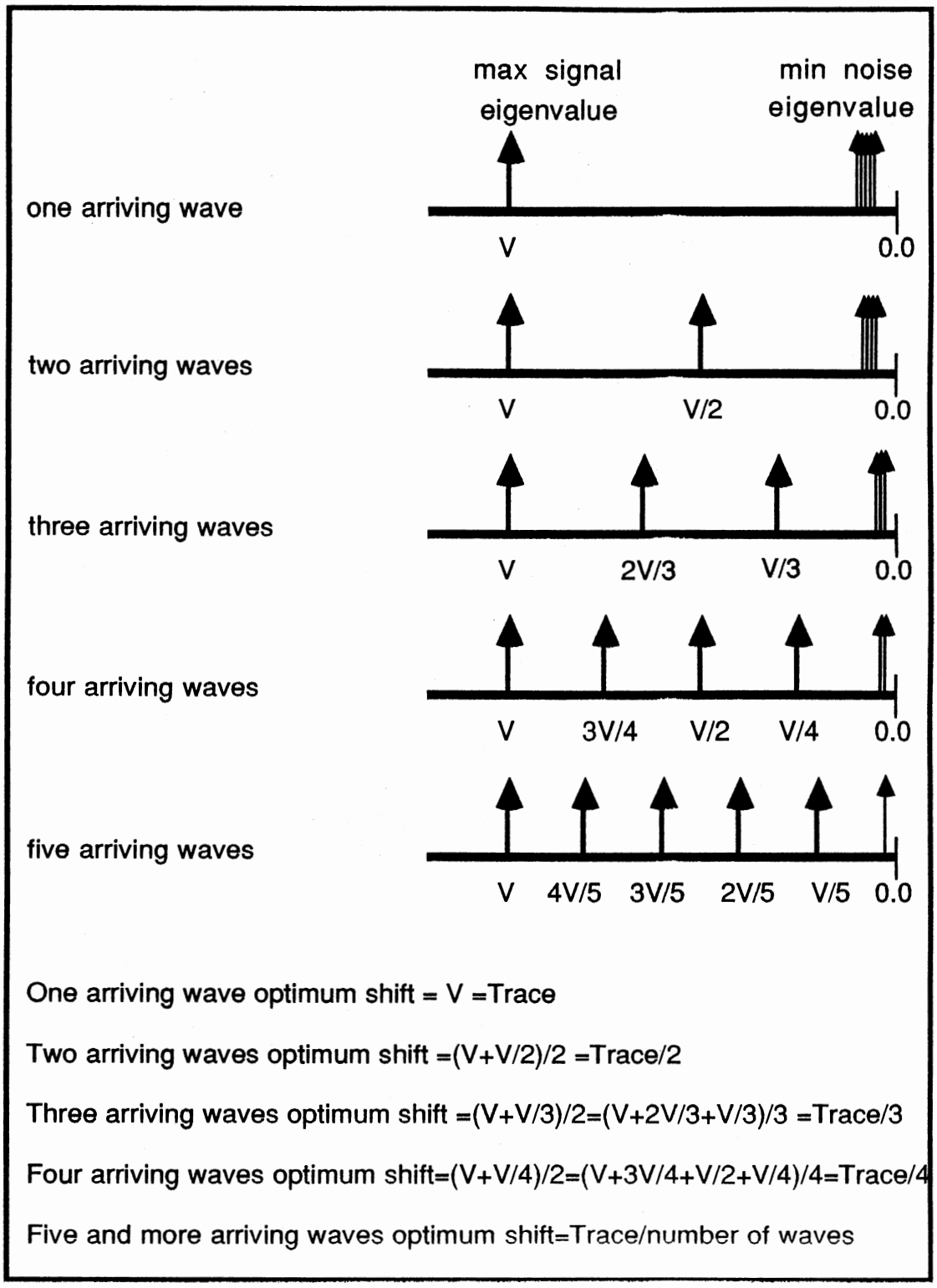


Figure 15. Determination of Optimum Shift for Eigenvalue Convergence

that limit the minimum number of iterations required for convergence which also limits the number of directions that are resolvable. These factors include the algebraic limit of the solution accuracy applied, the aperture width, the number of samples taken, the period of sample time, the true stationary nature of the sources, and the SNR of the situation among others.

As stated, if the number of arriving wavefronts is known, this would dictate the optimum shift value for the minimum eigenvalue search, which would cause the fast convergence. In this situation, this task would be completed in a straight forward manner. However, when the number of sources is unknown, the usual case, the serial approach using this technique is computationally extended over the parallel program. There are at least two basic approaches possible.

First, iterations could be completed for an estimate of one arriving wave before changing to the next estimate of two arriving wavefronts. That is, it would be required to build a shifted matrix based on one arriving wave, and continue iterations until the solution is within tolerance, or some threshold exceeded. The threshold selection is no simple task, as the minimum number of iterations required is dependent on the set of unknowns that developed the sample covariance originally. In any case, the next step would be to compute a new shifted matrix based on two arriving waves and once again complete iterations as before. It would be necessary to compare the number of iterations resulting for each estimate as the sequence climbs to more and more arriving waves. When the minimum number of iterations is determined, this would relate to the number of arriving waves as described above. The eigenvector resolved at that point would be the most accurate noise direction vector obtained.

A second approach would be possible if the computer memory capacity is

very large. It would be possible to shift an entire set of matrices, each shift based on a different estimate of the number of arriving wavefronts. Then each iteration each matrix would complete one cycle of the power method. The first matrix to converge would stop the process. This would sidestep the difficult threshold issue, and also require each shifted matrix to complete only slighly more than the minimum number of iterations, rather than the selected iteration threshold or the number for actual convergence depending on which came first compared to the other serial technique. However, for large antenna arrays, this method would significantly limit the number of possible arriving wavefronts that could be investigated because each shifted matrix would need to be continuously held throughout all computations in this serial procedure.

On the other hand, with the number of wavefronts still unknown, the parallel version that has multiple algorithms running concurrently has an unusual advantage during this task. This is because several different activities can be occurring simultaneously. The parallel computer is split into subsets of nodes operating concurrently at the first level of parallelization. The first subset of nodes will be assigned to locate the maximum eigenvector using no shift, as was recommended above. Next, there will need to be multiple node subsets concurrently using different shifts, derived from the Figure 15, searching for the minimum eigenset.

This splitting of the parallel computer into subsets could reduce the first level parallelization speedup parameters allowing fewer processors to be available for first level speedup. However, it will be found in the analysis of the later part of this chapter that the eigendecomposition first level parallelization shows the least efficiency of the four parallelized operations of the DOA problem. Also, since this task is a low multiple N-squared order, very little time is used in this operation as long as the number of iterations can be kept low. After four

processors are applied at the first level of parallelization for N as high as one hundred, very little additional computer time can be saved by adding more processors at the first level of parallelization. Hence sixteen processors can be most effectively be applied by simultaneously splitting the work into four or even eight separate algorithms with each using one of the shifts derived in Figure 15.

The shift that turns out to be optimum, will cause most rapid convergence to the desired eigenset, hence allowing termination of the processes. Clearly this provides an estimate for the number of arriving waves. If desired, additional processors could be added at this point allowing larger numbers of concurrent searches depending on the expected resolution of the particular system.

One additional feature is seen when too small of a shift amount is chosen due to significant over estimation of the number of arriving waves. In these cases, the iterations will converge to the maximum eigenvalue instead of the minimum eigenvalue, which will be immediately obvious by a sign change in the resulting eigenvalue. This is an additional computational bound that can be detected quite rapidly and places a limit on number of iterations in these cases. This provides a method to constrain the process on estimations of the higher numbers of arriving wavefronts, concentrating more processors at the first level of multi-processing on the lower number estimates of arriving waves when necessary.

Of course, the estimate of the number of arriving waves is necessary for use in the last operation, searching for the peaks, and this way an estimate can be obtained without extracting the entire eigensystem, an N-cubed order procedure. It also accelerates convergence providing the needed eigenvector in minimum time. For large arrays there is a significant savings in computer time used.

Figure 16 is the result of the number of iterations required considering nine

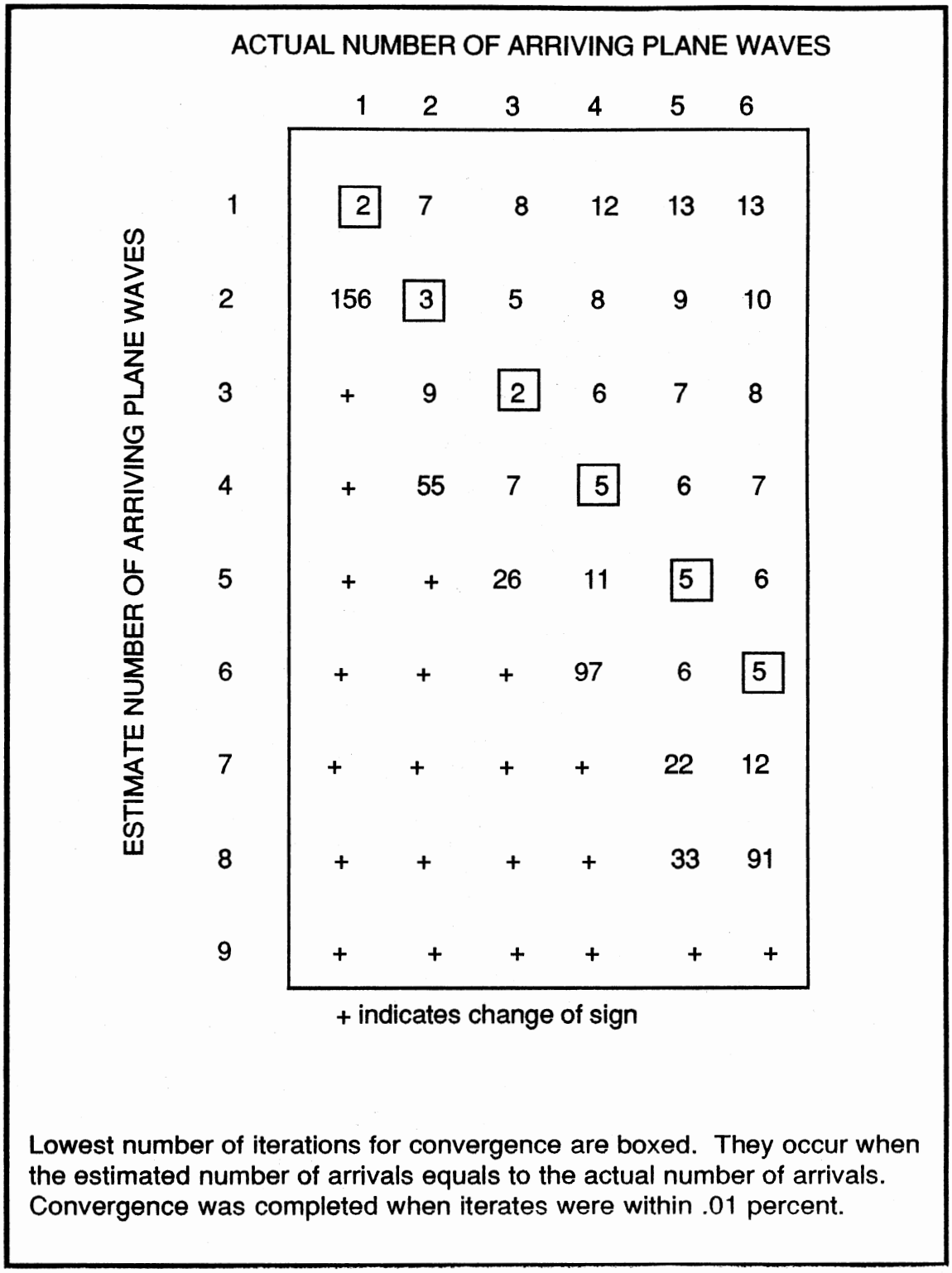
possible shifts for each of the sample covariance matrices using the values suggested in Figure 15. The boxed number of iterations is the minimum number of iterations that occurred each time. It can be seen that the estimated number of arriving wavefronts and the correct number correspond to the lowest number of iterations.

Figure 17 is a simplified flow diagram that shows the basic parallel events that occur concurrently during this operation. The specific data illustrated for this chart was extracted from Figure 16, the two source example. It can be seen that after three iterations, labled I =3, the node set , 8 - 11, would stop operation. The executive node of this set would then stop all of the other nodes that are also looking for the minimum eigenset. Of course, they are searching for the same minimum eigenset, but have a different estimate of M. Since nodes 0 through 3 were assigned the maximum eigenset search, they may or may not be complete depending on the particular situation. In this example it was true. All nodes recieve the estimate of M, the number of arriving wavefronts, and the maximum and minimum eigenvectors.

Not shown, but also important, were some examples where the wavefronts were too coherent and appeared as a single wavefront. In these cases the estimate of the number of wavefronts from this procedure also indicated one less source than was actually present.

In aliasing situations, where the antenna separation is greater than one half of a wavelength causing extra peaks, the estimate of M was found to be unaffected by the existence of the alias peaks.

It is now appropriate to analyze the power method to establish the mathematical model of its serial and parallel computational components. This provides an estimate of the parallel speedup possible and prepares for the parallel algorithm development and implementation.



Lowest number of iterations for convergence are boxed. They occur when the estimated number of arrivals equals to the actual number of arrivals. Convergence was completed when iterates were within .01 percent.

Figure 16. Results in Determining the Number of Arriving Waves

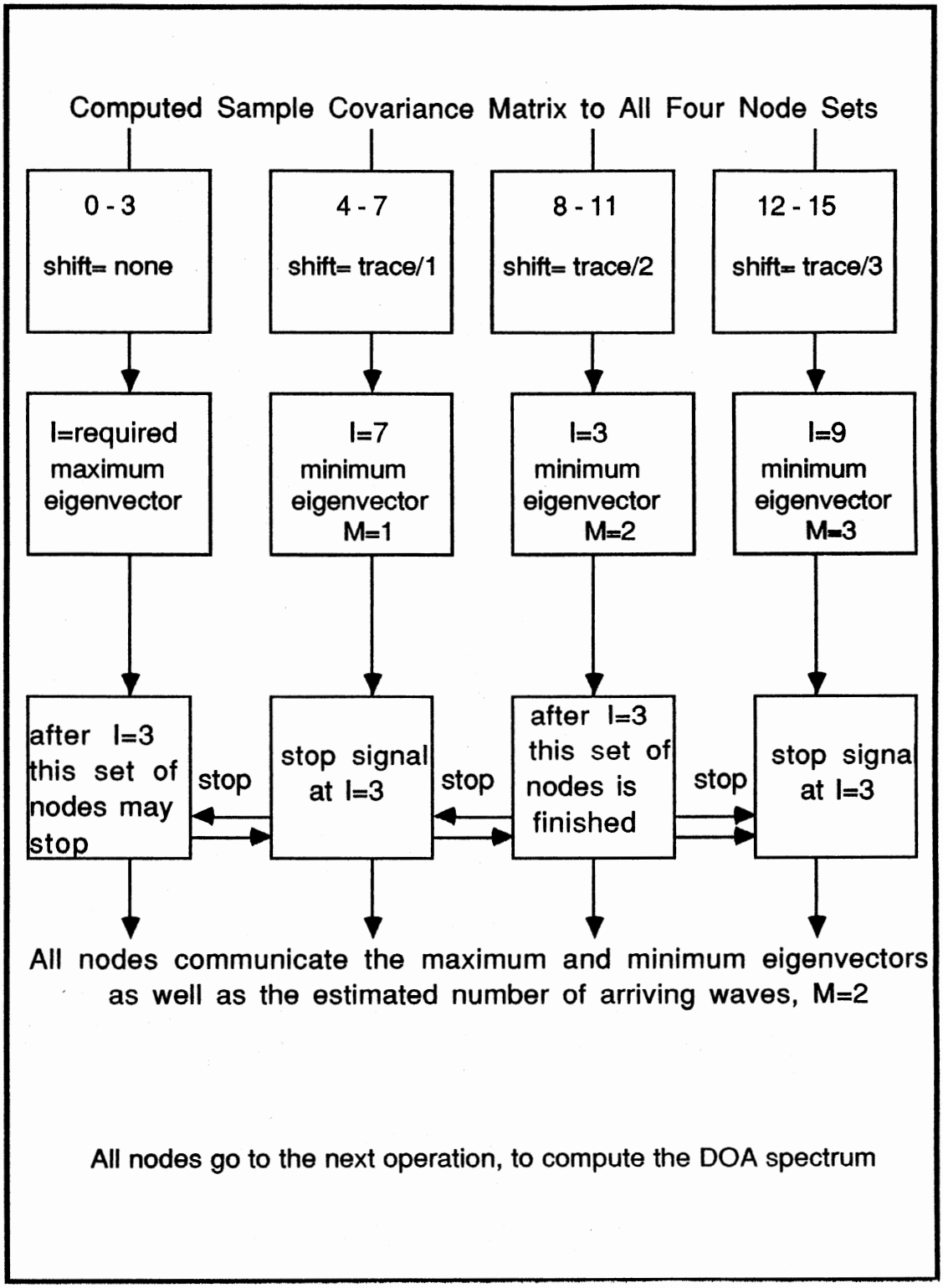


Figure 17. Simplified Flowchart of Multi-Algorithmic Acceleration

Power Method, Serial Instruction Count

Assuming a 2Nx2N matrix and I iterations, a straight forward analysis of the serial power method algorithm has shown that there are four computational steps in each iteration. Remember that N represents the number of antennas, however, since the elements of the sample covariance matrix are complex, the dimension N is multiplied by two.

The first step of the power method was the matrix-vector multiplication. This requires a total of $4N^2$ multiplications and $4N^2$ adds. Since these are computed within the same loop, register usage improves the performance over other sections so the average number of flops used will be represented as

$$5N^2 \text{ flops.} \tag{5-23}$$

The second step is the search of the computed vector for its maximum element. This step requires 2N-1 compares which will be equated to

$$2N\text{-}1 \text{ flops.} \tag{5-24}$$

The third major step requires the vector to be normalized by the maximum element located in step two. This would require 2N divisions, or

$$2N \text{ flops.} \tag{5-25}$$

In the final step the new vector must be compared to the previous vector, element by element, to determine if the convergence constraint has been satisfied. Instead, the alternative procedure of comparing the eigenvalues is used which will reduce computations required, especially in large arrays. Since the largest vector element is the maximum eigenvalue estimate, only one compare needs to be made, i.e.,

$$1 \text{ flop.} \tag{5-26}$$

If the new value is not close enough in accordance with the tolerance used,

then more iterations are required, or else the program ends with the maximum eigenset as the output.

A single shift in the matrix is expected in every minimum eigenvalue case. The shift amount is subtracted from the diagonal of the sample covariance matrix. The time consumed in computations for the normal eigenvalue problem shift will be just from the diagonal and equal to 2N additions.

Making the generalized eigenvalue solution shift requires the full noise covariance matrix estimate times the shift, to be subtracted from the sample covariance matrix. Therefore it requires $4N^2$ multiplications and $4N^2$ additions to shift the origin of the matrix before the decomposition begins. The shifts are not a function of the number of iterations, so they only occur once before beginning the iterations. As a compromise in the implimentation of these procedures, the second procedure was followed in all cases, being multiplied by zeros in the off diagonal cases for spatially withe noise.

There are also extra serial events, such as storing the resulting N dimensioned vectors, that occur. In these cases, a relativly significant amount of serial code is done when the iterations are very low. A simplified value for the serial model is provided that yields

$$I\,(5N^2+4N)+16N^2 \quad \text{flops}, \tag{5-27}$$

corresponding to I iterations using the power method .

Power Method, Parallel Instruction Count

The parallel power method using the first level of parallel speedup will take the same four steps, however the matrix will be split among the number of processors. The basic operation is a vector-matrix multiplication procedure. Each processor will compute its N/P element section of the new vector (again

assuming N/P is an integer which causes no loss in generality) by computing a vector-partial/matrix product. With P the number of processors being used in the parallel algorithm, then the first step requiring the vector-partial/matrix multiplication can be reduced to

$$5N^2/P \quad \text{flops.} \tag{5-28}$$

The P nodes would each require the entire vector, but only 2N/P rows or columns of the sample covariance matrix.

The second step, the search for the maximum element of the vector which is the estimate of the eigenvalue, can also be reduced in parallel by searching the computed segments of the new vector at each node concurrently. This requires

$$(N/P)-1 \quad \text{flops.} \tag{5-29}$$

This partial search will have to be followed by additional comparisons of the P distributed maximum elements to complete the localization of the maximum eigenvalue estimate.

The third step, normalizing the vector by its maximum element, cannot be performed at each node at this point because the overall maximum element will not have been determined from step two until the data arrives at the executive node and the determination between the set of P maximum elements is complete. The normalization will need to be accomplished in serial because the entire vector is needed at each node, hence no parallel savings can be obtained on this 2N-ordered computatlion.

The fourth step, comparing the present value to the last value, will require a single compare at the executive node. At this point, all of the participating nodes have sent their segment of the eigenvector from the computation of the vector-partial/matrix product, and their results of the partial searches for the maximum element to the executive node.

In summary, one node must act as an executive node for this process as well

as taking an equal share of the calculations. The executive node assembles the partial vectors computed at the other nodes into a complete vector. It compares the maximum elements for the largest single element which requires P-1 compares, and one more compare is required to check for convergence.

If the procedure has converged to the eigenvalue, it sends the current vector normalized by the eigenvalue, to the next operation, the third task that was defined in Chapter II. Otherwise, another iteration is required and the new vector is communicated to each of its working nodes. The nodes begin another iteration because they already have their required columns of the matrix.

Node zero, is used to monitor the overall activity of the multi-algorithms and when convergence from a node set with the optimum shift looking for the minimum is successful the data will be transmitted to this node. It is not necessary to have any of the executive nodes described, including node zero, to be in a wait state except for the instant after iteration, hence all nodes can be totally involved in the computations with negligible serial efforts.

Collecting all of the computations and comparisons above totals as follows:

$5N^2/P$	computations of the matrix-vector computation,
$(4N/P)-1$	compares to search for the largest vector at each node,
$P-1$	compares of the completed P partial compares,
1	check for convergence,
$2N$	divides to normalize the vector

These all sum to

$$((4N^2+4N)/P)+P+1+2N \quad \text{flops} \tag{5-30}$$

This is the total number of flops dedicated to the parallel computation during each iteration. Also an origin shift costing N/P flops, being split up among the P nodes, must be added to the parallel model. It is not a function of I, the number of iterations so it only is required once.

Because an MIMD message passing computer organization requires a message to be passed each time the nodes exchange data, $\log_2 P$ messages from the working nodes to the executive nodes are necessary to pass the computed partial values of the vector. Also one message from the executive node is needed to be distributed to each of its working nodes to begin the next iteration. Again, exact time required for the messages depends on their length and system interaction, so a single message time is represented by μ as the computer time used (in flops). Because the message is a function of the vector size, a factor of $\log_2 N$ needs to be multiplied times the established value of μ.

There still exist strictly serial noncomputational instructions that are required to be run, which are included in the computational load as was done earlier. This time needs to be accounted for at this level of approximation because the low number of computations completed when the iterations are only a few because of the acceleration procedures applied.

Finally, the total number of flops for the parallel algorithm for an NxN matrix taking I iterations can be estimated as:

$$I\left((5N^2+4N)/P\right)+16N^2/P+I\left(2\mu\log_2(P)\log_2(N)\right) \quad \text{flops} \qquad (5\text{-}31)$$

Notice again for Task two, that Equation (5-31), the parallel model, equals Equation (5-27a), the serial model, if the number of processors, P, equals 1.

Eigenstructure Decomposition Results

Table 2, the last page of this chapter, reports the speedup ratio and CTU using exactly two, twenty, and two hundred iterations to resolve the eigenset. This table was developed by allowing the estimator to run to the specified number of iterations using the given input set and processor situations. Comparisons between serial and parallel times is based on only the first level

of parallelization when considering the same number of iterations. The improvement possible when using multi-algorithmic acceleration as described earlier is not directly seen in the table. This is because it requires comparing different iterations needed against an unknown serial algorithm.
Due to the inefficiency in the computation versus communication workloads for small N, and the earlier stated serial computation activities, it is seen that increasing the number of processors actually decreases the speedup ratio for arrays of only sixteen elements. Here the parallelization threshold is exceeded.

In agreement with the speedup ratio, the CTU data shows that time is lost for values of N and P for small arrays, and a drop in improvement occurs with P greater than 4 for arrays of 16 antennas and high iterations. This indicates that when the number of antennas is small, this task of the parallel algorithm should use smaller numbers of processors for best efficiency and best speedup. This is what was mentioned earlier, and what has made the multi-algorithmic even more effective in this task. With large arrays and large numbers of iterations, the speedup differential improvement is quite small for increasing above eight processors, however a much larger decrease in time is seen when the number of iterations is reduced due to accelerated convergence.

This completes the first level analysis of the power method. The actual result gives the highest speedup ratio of about 13.47 with 16 processors, using a very large N of 160. In terms of speedup differential, this provided a savings of almost 132.98 seconds out of 143.64 seconds of the serial computer time used.

Notice however, when this same size array is reduced to just a few iterations and only two processors, the time is less than one fifth this best time achieved with high iterations and 16 nodes. The point being, that it is more significant time wise to reduce the number iterations required by splitting the problem and processors multi-algorithmically, than to apply the parallel processor in only a

first level parallelization effort. This is obviously the case in the smaller to medium size arrays because of the poorer speedup parameters that exist. In fact, an array of only 16 antenna elements reaches the parallelization threshold at only 4 processors and the speedup parameter is so low that it does not reasonably justify more than 2 processors.

It can be seen that this is also true for large arrays as long as the optimum scalar shift can be located and the lower number of iterations can be obtained. Further of course, the number of arriving wavefronts can be determined by using the multi-algorithmic procedure.

Actual DOA applications occur with all size arrays, so the large speedup improvements in this task indicates the higher end of possible real world gain. Smaller speedup ratios, and computer time savings associated with smaller antenna arrays still show worth-while improvement when it is considered that only a few processors will be applied. It is obvious that this is one point that tailoring the parallel system to the array would enhance the real system times over the compromise "same applies to all" type prototype implementation developed for this study.

As was stated earlier, the large savings attributed to reducing this procedure to an N-squared function by using the power method, and then using only the maximum and minimum eigenvectors, did not allow super high first level parallel improvement numbers for the speedup ratio and speedup differential to be obtained. The serial times are very very short relative to traditional SVD serial approaches using large values of N. The overall serial, parallel and multi-algorithmic parallel improvement of this chapter, however, allowed this research to reach the goal of a real-time DOA processor.

TABLE 2

EIGEN-DECOMPOSITION OF THE SAMPLE COVARIANCE MATRIX

TWO ITERATIONS

Number of Nodes	Number of Antennas 16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	0.040	1.00	0.616	1.00	1.37	1.00	3.82	1.00
2	0.024	1.67	0.312	1.97	0.70	1.96	1.92	1.99
4	0.024	1.67	0.168	3.67	0.35	3.91	0.98	3.90
8	0.024	1.67	0.088	7.00	0.19	7.21	0.49	7.80
16	0.024	1.67	0.064	9.62	0.11	12.45	0.27	14.15

20 ITERATIONS

Number of Nodes	Number of Antennas 16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	0.176	1.00	2.63	1.00	5.94	1.00	16.52	1.00
2	0.120	1.47	1.36	1.93	3.01	1.97	8.30	1.99
4	0.112	1.57	0.74	3.55	1.54	3.86	4.22	3.91
8	0.112	1.57	0.44	5.98	0.86	6.91	2.18	7.58
16	0.136	1.29	0.32	8.22	0.54	11.00	1.22	13.54

200 ITERATIONS

Number of Nodes	Number of Antennas 16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	1.504	1.00	22.90	1.00	51.69	1.00	143.64	1.00
2	1.056	1.42	11.78	1.94	26.14	1.97	72.18	1.99
4	0.976	1.54	6.38	3.59	13.50	3.83	36.69	3.91
8	1.080	1.39	3.86	5.93	7.52	6.87	19.04	7.54
16	1.264	1.19	2.84	8.06	4.73	10.93	10.66	13.47

Note: CTU is in seconds, S_ρ is the speedup ratio

CHAPTER VI

ARRAY MANIFOLD INTERSECTION

The third task required to be converted to a parallel algorithm is computation of the DOA function amplitude plot. This plot is computed by completing the inner products of the discovered noise eigenvectors, with the stored array manifold vectors which are based on the antenna array geometry. This is done in MUSIC by sweeping its peaking function, Equation (2-9), through all of the investigated values of possible incoming bearings using the entire set of estimated noise eigenvectors extracted from the sample covariance matrix. The function peaks where the DOA information contained in the eigenvector and the array manifold vector intersect. This task is highly computational intensive especially when only a few wavefronts are arriving. In these cases, since M is small, there are more N order dot products required for each angle investigated than in the large M case.

There was a dramatic computational reduction obtained in the previous chapter when it was assumed that only two vectors, those associated with the maximum and minimum eigenvalues, needed to be extracted from the sample covariance matrix. This chapter begins by validating that result, and provides the new two-eigenvector peaking function to be used in the parallel algorithm. Once again, as was found in Chapter V, a significant time savings will be gained over other EV/EV procedures using the two-eigenvector function.

In Chapter II it was noted that in the special situation when the number of

arriving signals is one less than the number of antennas, the smallest eigenvalue has an algebraic multiplicity of one, thus it has a distinct single eigenvector. This requires finding only the minimum eigenvector. In this case the MUSIC algorithm reduces to the same data as Pisarenko's harmonic retrieval method (Pisarenko, 1973, Schmidt, 1981). This fact indicates that in at least one case, it is correct to design an eigendecomposition algorithm based on the power method which finds one eigenset of a matrix at a time, through an iterative procedure (Conte, 1980).

Of course the smallest eigenset, not the largest, is needed for Pisarenko's method or when the noise subspace approach is being followed as in MUSIC. However, it has been discussed that with the simple modification of shifting the origin, the minimum (or noise eigenset) can easily be located while still maintaining the N-squared order of computation (Conte, 1980).

If the special case above can be generalized or enhanced, it may be possible to eliminate the need to have the set of eigenvectors $\mathbf{E} = [\underline{e}_{M+1}, \underline{e}_{M+2}, \ldots, \underline{e}_N]$ for the intersection computation with the array manifold. That is, in each case since a single eigenvector, $\underline{e}_{min}$, in the noise subspace exists that is orthogonal to the signal subspace, then the peaking function could possibly be reduced to using just this vector. Further, it would then be possible and efficient to apply the power method to find this single eigenset. This would result in replacing the MUSIC peaking function, Equation (2-9), for operation three with the following:

$$m(\theta) = \frac{1}{\underline{a}^H(\theta)\, \underline{e}_{min}\, \underline{e}_{min}^H\, \underline{a}(\theta)} \cdot \qquad (6\text{-}1)$$

This would, in turn, reduce the number of dot products compared to MUSIC across N-M eigenvectors to only one eigenvector. This effectively reduces the order of computations from N-cubed to N-squared. It was stated that the

function will eventually use two vectors, not one, but first the conditions which allow a single minimum *noise direction vector* to exist in $\mathbf{R}_x$ will be detailed.

Direction Eigenvectors

Solving the eigenstructure problem for the single eigenvector $\underline{e}_{min}$, where $\underline{e}_{min}$ is the noise direction vector of the eigenset related to the smallest eigenvalue of the noise subspace, can be done as an iterative task using the power method. The discovered vector will be shown to lie in the subspace spanned by the set of eigenvectors $[\underline{e}_{M+1}, \underline{e}_{M+2}, \ldots, \underline{e}_N]$.

First it is necessary to establish the independence of the generalized set of eigenvectors, all associated with the same eigenvalue λ_{min}. It has been seen that in this problem, with an appropriate shift to the matrix, the minimum eigenvalue can become the maximum eigenvalue. Hence, any of the following that is developed for the maximum eigenvalue also directly applies to this minimum eigenvalue problem.

The eigenvalue, λ_{min}, will have multiplicity (N-M) established in Chapter II. The defining equations for the set of vectors $[\underline{e}_{M+1}, \underline{e}_{M+2}, \ldots, \underline{e}_N]$ all associated with λ_{min}, where $\underline{e}_{M+1}$ is a unique eigenvector and the rest are generalized eigenvectors all associated with the same eigenvalue λ_{min} are:

$$\mathbf{R}_x \underline{e}_{M+1} = \lambda_{min} \mathbf{R}_b \underline{e}_{M+1}, \quad \underline{e}_{M+1} \neq \underline{0} \quad \text{or} \quad (\mathbf{R}x - \mathbf{R}_b\lambda_{min})\, \underline{e}_{M+1} = \underline{0} \qquad (6\text{-}2)$$

$$\mathbf{R}_x \underline{e}_{M+2} = \lambda_{min} \mathbf{R}_b \underline{e}_{M+2} + \mathbf{R}_b \underline{e}_{M+1} \quad \text{or} \quad (\mathbf{R}x - \mathbf{R}_b\lambda_{min})\, \underline{e}_{M+2} = \underline{e}_{M+1} \qquad (6\text{-}3)$$

$$\mathbf{R}_x \underline{e}_{M+3} = \lambda_{min} \mathbf{R}_b \underline{e}_{M+3} + \mathbf{R}_b \underline{e}_{M+1} \quad \text{or} \quad (\mathbf{R}x - \mathbf{R}_b\lambda_{min})\, \underline{e}_{M+3} = \underline{e}_{M+2} \qquad (6\text{-}4)$$

$$\vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots$$

$$\mathbf{R}_x \underline{e}_N = \lambda_{min} \mathbf{R}_b \underline{e}_N + \mathbf{R}_b \underline{e}_{N-1} \quad \text{or} \quad (\mathbf{R}x - \mathbf{R}_b\lambda_{min})\, \underline{e}_N = \underline{e}_{N-1} \qquad (6\text{-}5)$$

From the Equations (6-2) and (6-3), $(\mathbf{R}_x - \mathbf{R}_b\lambda_{min})^2 \underline{e}_{M+2} = (\mathbf{R}_x - \mathbf{R}_b\lambda_{min})\underline{e}_{M+1} = \underline{0}$. Multiplying Equation (6-4) by $(\mathbf{R}_x - \mathbf{R}_b\lambda_{min})^2$ gives

$(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^3\,\underline{e}_{M+3}=(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^2\underline{e}_{M+2}=\underline{0}$. In general, it can be seen that $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^p\,\underline{e}_{M+p}=\underline{0}$, and $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^{p-1}\,\underline{e}_{p=}\,\underline{e}_{M+1}$. Since $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^N=(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^{N-p}\,(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^p$, $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^N\,\underline{e}_{M+p}=\underline{0}$ for p=1,2,. . . , N-M, it shows that all of these vectors belong to the null space of $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^N$ (Brogan, 1985).

Further, it can easily be shown that these vectors will be linearly independent. This is a classical result, but repeated here to bring it into the context of the minimum eigenvector set which makes up the noise subspace. Let,

$$a_1\underline{e}_{M+1}+a_2\underline{e}_{M+2}+\ldots+a_{N-M}\underline{e}_N=\underline{0}\,. \tag{6-6}$$

It can be shown that this implies that each $a_i = 0$, which shows linear independence.

The first step is to multiply Equation (6-6) by $(\mathbf{Rx}-\mathbf{R}_b\lambda_{min})^{N-1}$. This gives $a_{N-M}(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^{N-1}\,\underline{e}_N=\underline{0}$. But from above $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^{N-1}\,\underline{e}_N=\underline{e}_{M+1}\neq\underline{0}$, so for this case a_{N-M} must be zero. It follows from this fact that if Equation (6-6) was multiplied by $(\mathbf{R}_x-\mathbf{R}_b\lambda_{min})^{N-1}$, then likewise $a_{N-M-1}=\underline{0}$. Continuing this process over all i from 1 to M is $\Sigma\, a_i\underline{e}_{M+i}=\underline{0}$, then $a_i=0$ for i = 1,2,...,M. Hence, the set of generalized eigenvectors $[\underline{e}_{M+1}\,,\,\underline{e}_{M+2}\,,\ldots,\underline{e}_N]$ is linearly independent (Brogan, 1985).

If there are a number of independent eigenvectors corresponding to a repeated dominant eigenvalue (minimum, made dominant in this case), this does not affect convergence of the power method and the iterates tend to some vector lying in the subspace spanned by the eigenvectors (Wilkinson, 1968). This eigenvector will be a linear combination of the eigenvector set and corresponds to the multiple maximum eigenvalue (Gourlay, 1973).

This is analogous to the work of Kumaresan and Tufts (1983) where they developed a polynomial D(z) whose zeros fall near the noiseless locations in a

moderate SNR situation. They resolve a single vector $\underline{d}$ in terms of the noise subspace eigenvectors, whose values are the coefficients of the polynomial. Their method of resolution of this single vector does not eliminate are greatly reduce the computational burden, and they indicate that the resulting procedure is essentially the same as that of Reddi's (1979).

Here, with the noise mean assumed near to zero, the larger the number of samples, the closer the smallest eigenvalues become. There will be a distinct eigenvector and N-M-1 independent generalized eigenvectors. It is possible to locate the absolutely smallest eigenvalue and its associated eigenvector using the power method. The eigenvector resulting will be the minimum noise direction vector, $\underline{e}_{min}$, necessary for application in Equation (6-1).

The problem with using a single noise direction vector is that it is greatly affected by the noise, and additional peaks appear that are not actual angles of arrival. Figure 18 is an example of such a situation. It is the plot derived by using 16 antennas with .5λ spacing and three arriving waves at 25, 30, and 35 degrees with a 10 dB SNR. The three directions are indicated at their correct values by peaks, but because there are many other peaks that have higher amplitudes, the actual wavefronts can not be identified. Hence, in this case, these spurious peaks make it impossible to extract the three actual arriving wavefronts.

This problem was discovered by Schmidt and is why his algorithm MUSIC uses all of the noise eigenvectors which effectively averages out the unwanted extra peaks. It is also related to the problem of the MUSIC algorithm generating the number of peaks estimated, correct or not. The problem for this research is that EV/EV techniques on this operation and in the last operation require an excess number of computations to extract the eigensystem.

This dilemma is uniquely resolved by additionally solving for and applying in

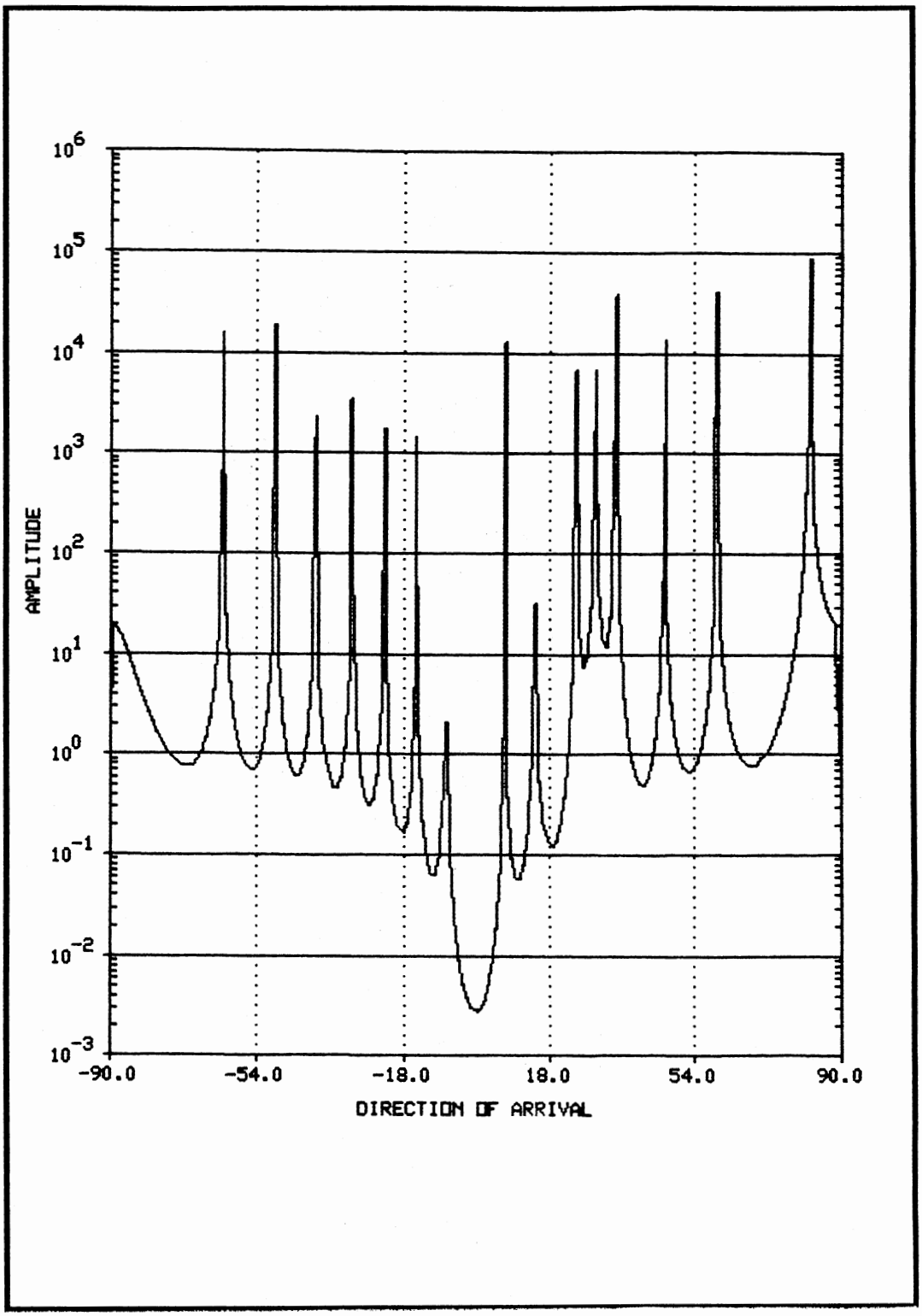


Figure 18. Plot of Minimum Noise Eigenvector Function, $m(\theta)$

a new two-vector peaking function the *maximum signal direction vector.* This is one of the contributions of this research.

Considerable work in direction finding has also centered on the set of maximum eigenvectors, the signal eigenvectors. A study by Reddi (1979) showed the maximum eigenvectors could be applied in a relation as the principal polynomial function where the roots caused the function to vanish at the angular locations coinciding with the DOA. He recognized the high resolution difficulty when closely spaced signals coalesced into one source. He further recognized increasing the power levels of the sources or the aperture by increasing number of elements resulted in more sources being resolved. The maximum eigenvalues were shown however to also provide an unbiased estimate and that it was very accurate without spurious peaks. This was essentially a signal subspace complement to the MUSIC algorithm using all of the signal eigenvectors.

The maximum signal vector can be located beginning with the identical inputs as the minimum noise vector as described in Chapter V. In fact, as described there, it is accomplished simultaneously using a multi-algorithmic approach by splitting the parallel processor among tasks.

Simplifying Reddi's approach, in a Pisarenko-like effort, this research applies only this single largest eigenvalue's eigenvector, called $\underline{e}_{max}$ here, in a function which relates the intersection of the maximum eigenvector of the sample covariance matrix and the swept array manifold. This function will peak when the estimated direction of an incoming wave coincides with the correct bearing which is contained in the sample data. This peaking function uses the same **A** matrix elements as in Equation (6-1), and is as follows:

$$g(\theta) = (\underline{a}^H(\theta)\, \underline{e}_{max}\, \underline{e}_{max}^H\, \underline{a}(\theta))^2. \qquad (6\text{-}6)$$

A typical output for this function is very smooth and resembles the classical

beamforming methods which require large apertures compared to the noise eigenvector methods to provide similar resolution capabilities.

Figure 19 is an example of the the outputs using the same inputs as were used for Figure 18. It can be seen that for the range of values that the actual signals are arriving, approximately 20 degrees to 40 degrees, a peak spread from an amplitude of 1 to above 10,000 exists, with the peak at 30 degrees. There is a considerable amount of sidelobe activity from the antenna array, however, and the levels of output for almost all of the sidelobes are less than one in this example. Hence, the three arriving signals coalesced into a single peak centering at 30 degrees and either a greater aperture or improved SNR would be necessary to resolve the three arriving wavefronts.

The function g(θ) computed with the maximum direction vector is a squared value to enhance the emphasis on the detected arriving waves, amplitude greater than one, and at the same time de-emphasize the unwanted sidelobe activity, which effectively further filters unwanted peaks.

At this point, it can be seen that a proper combination of these two eigenvector outputs will provide an enhanced DOA estimation. It will be better than either alone. Equation (6-7) combines the information in the two functions using the same array manifold vector input and both the maximum and minimum eigenvectors. Hence, the final DOA function derived is as follows:

$$\xi(\theta) = g(\theta)\, m(\theta) = \frac{(\underline{a}^H(\theta)\, \underline{e}_{max}\, \underline{e}_{max}^H\, \underline{a}(\theta))^2}{\underline{a}^H(\theta)\, \underline{e}_{min}\, \underline{e}_{min}^H\, \underline{a}(\theta)} \qquad (6\text{-}7)$$

In summary, Equation (6-7) is computed for each azimuth bearing or portion of angle to be investigated. Where peaks in this discrete plot of values appear, can be interpreted as the possible DOAs. The peaks occur because the DOA function, Equation (6-7), peaks sharply for the zeros, or near zeros in the

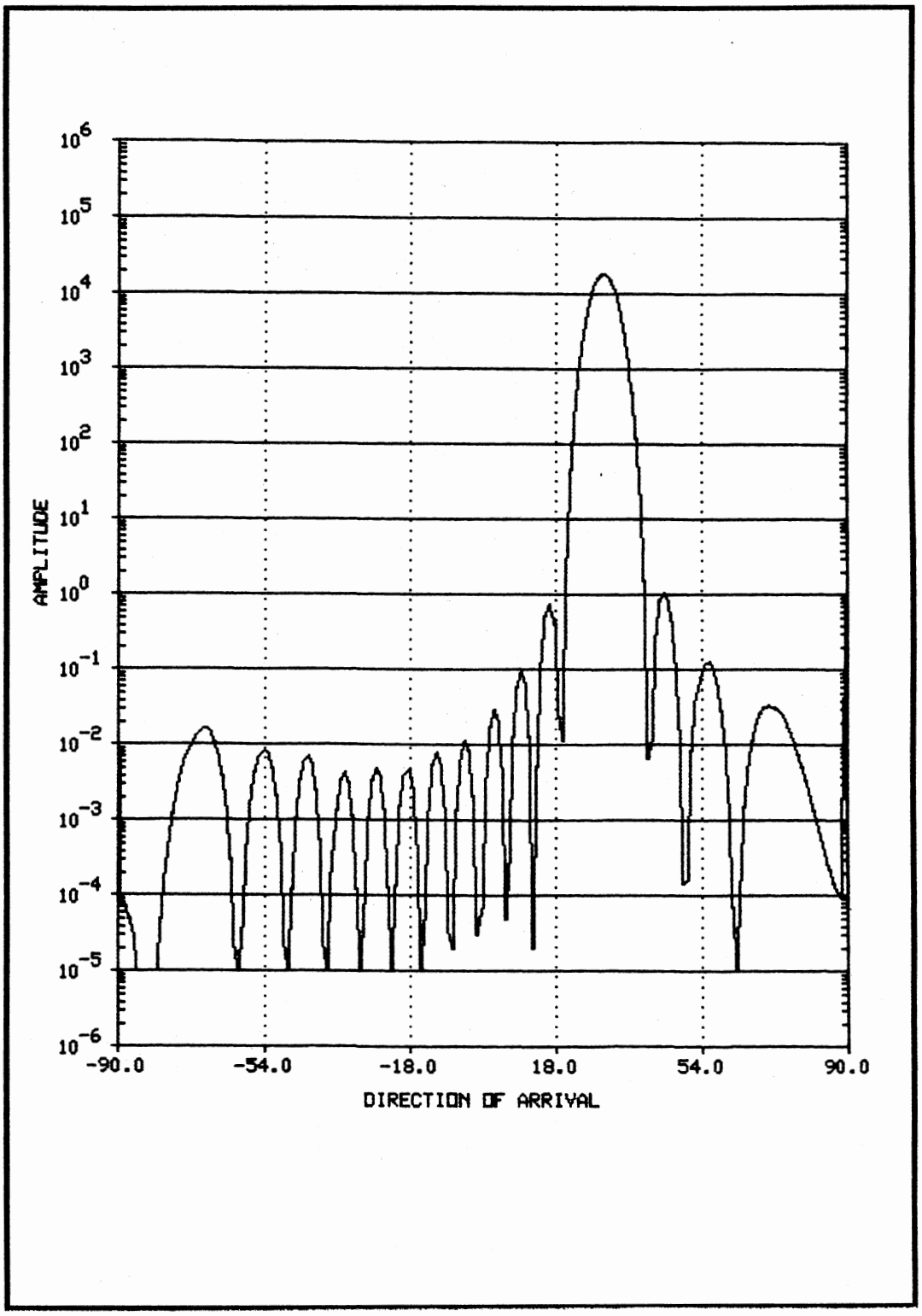


Figure 19. Plot of Maximum Signal Eigenvector Function, $g(\theta)$

denominator due to the orthogonalities of the single minimum eigenvector, $\underline{e}_{min}$, and the array manifold components, $\underline{a}(\theta)$. However, the new DOA function is also conditioned by the effects associated with the single maximum eigenvector, $\underline{e}_{max}$, and the array manifold components, $\underline{a}(\theta)$.

The final plot, Figure 20, shows the results using Equation (6-7) to compute the possible DOAs using the same simulated wavefronts as Figures 18 and 19. Here, there is no longer ambiguity between the extra peaks due to with the minimum eigenvalue component, and also, there is no coalescing of the closely spaced wavefronts due to the maximum eigenvector component. Therefore, this new combined two-eigenvector function yields the best features of each by using both ends of the eigensystem. This unique min-max vector approach has the effect of finding the maximum SNR events and relates them to the DOAs. An anologus function is used in beamforming when working with adaptive array systems, except that it requires an eigenstructure decomposition of the sample covariance matrix and knowledge of the noise matrix (Monzingo, 1980).

Using the DOA function, Equation (6-7), for implementation, the number of computations will be resolved to investigate the speed of computation.

Serial Array Manifold Intersection Instruction Count

Each dot product is a N order function. Again, because complex values are used, 4N products and 2N sums are required for the denominator, and a similar number for the numerator. Hence,

$$12N \quad \text{flops} \tag{6-8}$$

are necessary for each portion of an angle investigated. The computation of the function $g(\theta)$ results in a scalar that needs to be squared, and then multiplied by the computed value of $m(\theta)$, another scalar. These individual

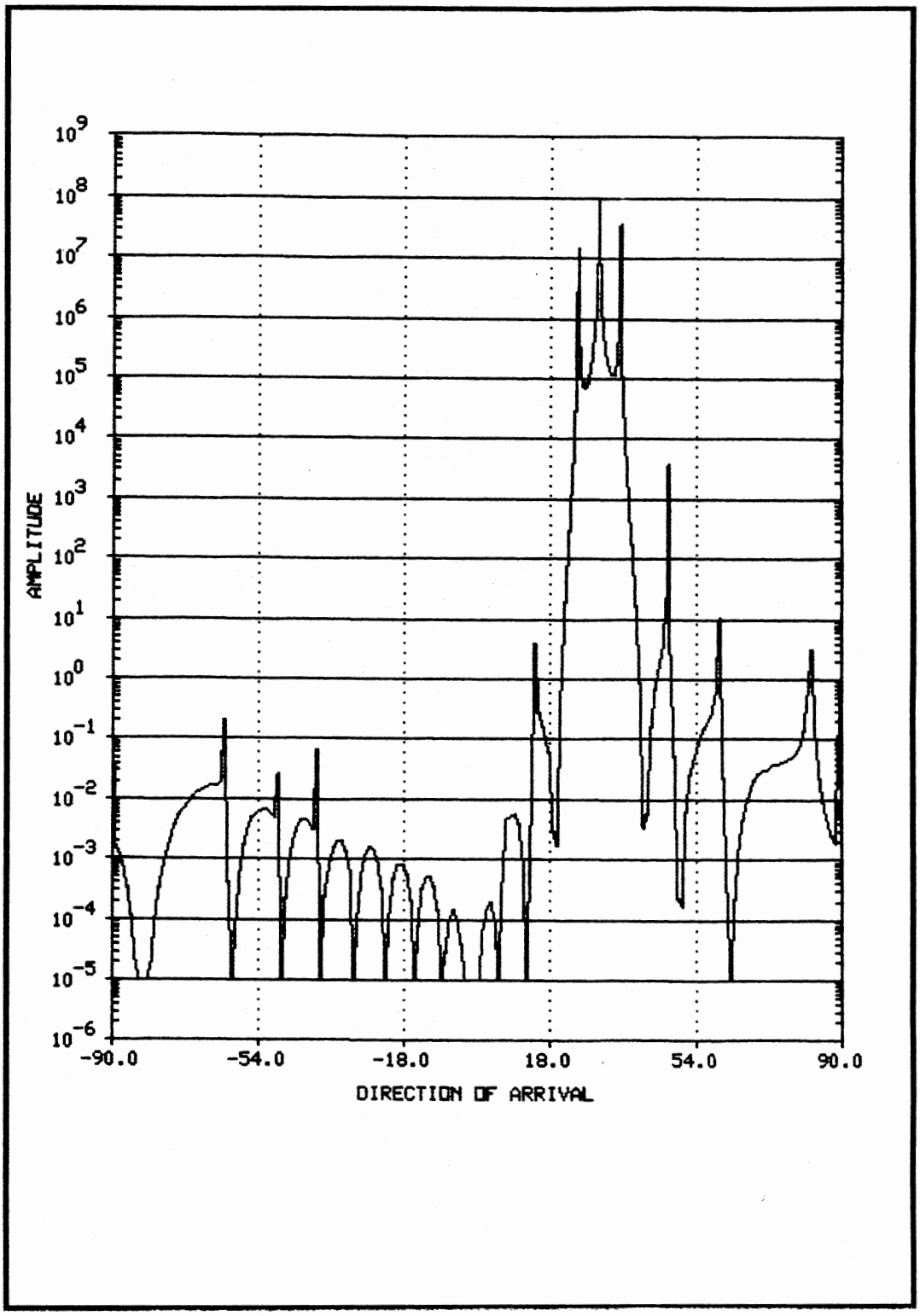


Figure 20. Plot of Combined Two-Vector Estimator Function, $\xi(\theta)$

computations will be neglected compared to the multiple N ordered computations of the dot products. Using D as the number of angles, or portions of angles investigated, the final equation to approximate the serial computations will be,

$$D12N \quad \text{flops}. \tag{6-9}$$

As before, with the serial model complete, the parallel model will be developed.

Parallel Array Manifold Intersection Instruction Count

Each of the computations in the function of Equation (6-7) are independent of the arriving angle, θ_m, or the number of arriving signals, M, with the total number of computations depending only on the resolution needed to discover the DOA peaks, and N. The computational independence of these operations indicates that the serial time can be reasonably expected to be reduced by the order of the number of processors applied. There is no time needed for the internodal communications of the input because it was modeled to be left in place in the last chapter. The combining of the information will be left until the next operation is completed, the search for the peaks, so no time is used in this operation for internodal communications.

In parallel then, still using P processors, the solution will split the investigation of the possible DOA bearings between the P processors taking:

$$(12DN/P) \quad \text{flops}, \tag{6-10}$$

to compute the required inner products searching for orthogonality and peaking conditions of the eigenvector and array vectors for each value of investigated DOA. The value of D will normally be a minimum of 1800 for an azimuth search corresponding to one tenth of degree resolution for a single elevation. In this

case, there is negligible noncomputational strictly serial processing and no value was included in the model.

Array Manifold Intersection Results

Table 3 is based on a value of 1800 for D, representing one tenth degree resolution in azimuth. Division of the computations is among the investigated elevation angles, sweeping the function in each node its partial range of the values of D. The data for only a single arriving wave has been presented since the computations are not a function of the number of arriving waves.

It should be noted that an additional time delay due to the last operation of locating the DOA is included in Table 3 data. The last operation is integrated into this table time wise because the investigation of the computed values for peaks is accomplished as each of the values are calculated. This makes it unnecessary to store the resulting data saving storage space. Also, making the comparison as it is computed is a more effective approach because it eliminates the requirement to later search and read the all of the computed data. The next chapter discusses Task 4 activity in detail, but the additional portion of computations are of order D with a single arriving wavefront and is negligible time wise compared to the computational burden of this task even with the smallest N used in these experiments.

Table 3 data also includes the internodal communications time resulting from the last operation, however it to is very small and does not impact the results in a measurable way.

The CTU data shows significant amount of time decrease as a result of adding processors. This indicates the worth-whileness of using parallel processing on this operation (and the next). This is not surprising because the

message processing load is very small and the large number of computations in both operations are independent. The parallel efficiency of over 91 percent using 160 antennas and 16 nodes was the highest obtained value for any of the other operations. The greatest time saving occurs at the largest value of 160 antennas and 16 nodes, saving 20.13 seconds out of 21.16 seconds of serial computer time used.

The mathematical simplicity and mathematical directness of this task, makes it not vulnerable to a multi-algorithmic attack. But actually, non-cooperating multi-data procedures are used for these computations, because the processors are using different data sets for the different sweeps. No further speedup has been obtained by communication betweem these algorithms.

TABLE 3

COMPUTATION OF DOA AMPLITUDE PLOT
AND BEARING LOCALIZATION

ONE ARRIVING WAVE								
Number of Nodes	Number of Antennas							
	16		64		96		160	
	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ	CTU	S_ρ
1	2.104	1.00	8.58	1.00	12.93	1.00	21.61	1.00
2	1.056	1.99	4.30	1.99	6.47	1.99	10.83	1.99
4	0.552	3.81	2.19	3.92	3.29	3.93	5.48	3.94
8	0.288	7.31	1.11	7.73	1.66	7.79	2.75	7.86
16	0.160	13.15	0.60	14.30	0.90	14.37	1.48	14.60

Note: CTU is in seconds, S_ρ is the speedup ratio

CHAPTER VII

DOA INVESTIGATION

The final task is searching the discrete plot that resulted from computations in Chapter VI for the peaks which represent possible DOA bearings. As stated earlier these computations are completed when each of the plot element values are computed from Equation (6-7). Although integrated into that step, It will still be valuable to provide an analysis of the serial and parallel models to estimate the amount of time expended during this task.

Serial Instruction Count for DOA Investigation

The number of computations in the serial search for a single arriving wave using D possible arrival bearings is simply,

$$D\text{-}1 \quad \text{(flops)} \tag{7-1}$$

for the compares looking for the single largest value.

For M multiple wavefront arrivals, the estimated number of arriving waves from Chapter V will determine how many sorts and the number of maximum peak locations that will be necessary to be located. Knowing the number of arriving signals was required because of the peaking nature of the noise eigenvector portion of the estimator function. Extra peaks will exist at lower levels often not to far from the magnitude of the actual peak values. The estimation method for M developed earlier is necessary in this operation to rule out the additional peaks as possible signal sources.

Considering the highest number of computations necessary, will show M compares for each peak that exists in the computed plot. There will be approximately one peak for each half wavelength separation set used, however most of their values will be very low compared to the actual magnitude values of real peaks due to the function using the two-eigenvectors. Hence assuming a reasonable threshold can be set, the DOA search requires little more than M sorts, M times, as the function is swept through D possible bearings. This yields a serial total of,

$$DM^2 \text{ (flops)}. \qquad (7\text{-}2)$$

If M were large then additional sort procedures could be applied which would reduce the M2 term. In actual practice, M is much lower then N, and rarely exceeds ten, hence this is a very small component in this problem.

The next step is to make the parallel mathematical model analysis of this operation, before going on into the overall model development.

Parallel Instruction Count for DOA Investigation

The parallel version of this task begins within each of the processors as the computations are being made. The comparisons of the largest peaks found (if any) are made within the P processors against the threshold used. There was no need to distribute the data to these nodes because it is being used as it is being computed in the last operation. After Task 3 is completed on each node, and each node has investigatd its data for peaks, there will be a final $\log_2(P)$ messages and at most $M\log_2(P)$ comparisons of the distributed data as it is reassembled and analyzed at the executive node to find the M maximum peak values indicating the M different DOA estimations. These messages between the P processors each take a minimum of μ flops of time. Compared to the

serial search, the parallel algorithm yields:

$$DM^2/P + (\mu+M)\log_2 P \quad \text{(flops)} . \qquad (7\text{-}3)$$

Searching for the peaks is not dependent on the value of N , the number of antennas, or S, the number of samples per antenna. The value for M is usually very small compared to N, although it may theoretically be N-1 for N independent antennas. This limit is not possible using large arrays with the system implemented, and the largest number of arriving wavefronts simulated was 10.

No additional time in this operation was measured between a single wavefront sweep or the 10 wavefront sweep using 64 antennas and 16 nodes, hence no data was tabulated for this operation for variable M. At the smaller end, the times between 1 or 10 wavefronts will be relatively larger compared to the last operation, but still not of a measureable concern. In any case, the time consumed for this operation is very small and the results were included in Table 3 as stated in Chapter VI.

This operation takes relatively the least amount of time of the four tasks, especially when S and N are large values. However, for a three dimensional search using other than a colinear array as described in Chapter one, a ten degree elevation search (also with one tenth of a degree resolution) would immediately increase this computation level one hundred fold. As more degrees of elevation, or finer resolution in both directions are required, then more time would be expended in this task. But this increase will also apply to the previous task which is a function of D times N. Therefore, this task can never dominate the overall process except for very small N and S where the times are already in the real-time arena.

Increasing D will improve the computation to communication ratio, and would improve the speedup ratio for all values of P. Since all of the

computations are done in place, and only the estimated M DOA values are needed to be forwarded in the final internodal messages, it is seen that this task also closely approximates a perfectly parallel process. It can therefore be expected to improve by nearly the order of P when put into the parallel form.

The exceptional sharpness (high resolution) of the peaks is the major contributor to the problem of not being able to pick up information about the location of the peaks earlier than when the computation is within a few degrees of the estimated bearing. In fact, the more precise the eigenvector and the better the signal to noise ratio, the finer the investigation must be extended to discover the peak. Also, since the function peak height is not directly correlated with the power in the signal, and only roughly to the SNR between arriving signals, it is not possible to predict the magnitude height of the peak to assist in its localization.

CHAPTER VIII

OVERALL SPEEDUP ANALYSIS

All four tasks have been analyzed and new serial and parallel algorithms have been accomplished for each task. Timing models and the individual operation results were provided within each chapter. It was seen that each operation has its own optimum value for number of processors that should be applied depending on the values the following parameters:

N, the number of antennas,

S, the number of samples,

I, the number of iterations required,

D, the number of investigated angles, and

M, the number of arriving waves.

Thus, even when considering only first level parallel speedup procedures, it is important to balance the efficiency and speedup parameters to select optimum number of processors to apply at each stage when using the smaller numbers represented by these parameters. As the parameters reached rather large values, it was seen that combining first level speedup with cooperative multi-algorithmic speedup in the eigendecomposition still required an informed choice of the number of processors to be assigned to reach the optimum overall speedup value.

The following model will be a unified combination of the previous four chapters' individual operation mathematical models. Actual overall results will

also be presented to make comparisons and reach some overall conclusions as to the overall speed performance of the parallel DOA estimator that is based on only the maximum and minimum eigenvectors.

The actual results will be limited to two iterations in the parallel mode because a single arriving wave is used. This same two iteration maximum is applied to the serial simulations. However note that this would be incorrect in the cases where more than one arriving wavefront is expected. However, since the actual number of iterations is highly system and data dependent, the improvement in reduction of iterations can not be quantified between the serial and parallel procedures.

Therefore, even though the maximum times in the parallel modes are accurately represented, it should be understood that the serial times obtained may be significantly lower than actual values if the iterations were not artificially terminated for comparison purposes.

This affects only one component of the overall model, the eigenstructure decomposition. However, in the worse cases for the serial mode, it could represent a large variation in the serial time. This would provide a much larger speedup ratio parameter which could cause the overall parallel performance to exceed 100 percent efficiency. However, because the CTU for the parallel cases are reasonably represented by this limiting iteration assumption, the lack of worst case serial performance will not be addressed.

There is also a shortage of published data to compare the parallel computer times obtained in this research, and the times obtained in the conventional EV/EV system research. However, some indicators of performance have already been stated, and it should be sufficient to point out that the reduced order of the operations obtained will result in approximately an improvement in the order of (NP).

Overall Model

As a first order approximation to an overall model, the eight speedup ratio and differential equations of the earlier chapters are combined in this section into two equations. The equations combined to compute the expected overall speedup parameters are Equations (4-4b), (4-10b), (5-27), (5-31), (6-9), (6-10), (7-2), and (7-3).

Equation (8-1) is the computed overall serial model based on the summation of Equations (4-4b), (5-27b), (6-9), and (7-2).

$$\{4SN^2\}+\{I\ (5N^2+4N)+16N^2)\}+\{12DN\}+\{DM^2\} \quad \text{(flops)} \qquad (8\text{-}1)$$

Equation (8-2) is the computed overall parallel model based on the summation of Equations (4-10b), (5-31), (6-10), and (7-3).

$$\{4SN^2/P+\log_2P(400+N^2)\}+\{I\ (5N^2+4N)+16N^2))/P+2\mu\log_2(P)\log_2(N)\}+$$
$$\{(12DN/P)\}+\{DM^2/P+((M+\mu)\log_2P)\} \quad \text{(flops)} \qquad (8\text{-}2)$$

The overall speedup ratio is Equation (8-2) divided by Equation (8-1).

The overall speedup differential is Equation (8-1) minus Equation (8-2) times the time for one flop. Equation (8-2) times the time for one flop is equivalent to the CTU

Overall Results

Table 4, located as the last page of this chapter, provides the actual measured times resulting from the computer simulations. It is essentially the combination of the first three tables. The data represents the single arriving wavefront case with D equal to 1800, representing one tenth of a degree bearing resolution from $-\pi/2$ to $\pi/2$ radians. The table data also includes time for only two iterations, because of the improved estimation situation associated

with a single arriving wavefront. Additional columns and tables could be formulated for other variations, but this sdingle table is sufficient to demonstrate the overall system performance obtained.

Depending on the size of the antenna array, the on-line or real-time capability begins at 96 antennas with 32 samples and 16 nodes, 64 antennas with 64 samples and 16 nodes, or with 64 antennas with 160 samples and 16 nodes.

In terms of near real-time performance, it is seen that using 16 nodes reduces every expertiment time recorded to an overall CTU less than 10 seconds. The longest time with 160 samples and 160 antennas has a CTU of 8.71 seconds on 16 nodes. In comparing the improvement against the N-squared serial algorithm, it is seen that the serial run time slightly exceeds two minutes. This is a speedup differential of 13.81.

Of course these results apply to the improved serial algorithm developed within this research. CTU times for the N-cubed based EV/EV algorithms could be expected to exceed an hour or more of computer time with this size of an antenna array. From the analysis completed earlier, and considing the times published for processing of antenna arrays up to16 elements, leads one to the conclusion that the times reported here are among the fastest possible on anything other than a supercomputer facility.

When 160 antennas were used, it was not possible to reach on-line times with 16 nodes. Extending the work to 32 nodes, or even 64 nodes, should achieve real-time performance with 160 antennas. A reasonable method to approximate the times of the extended dimensioned computer can be made by using the overall mathematical model developed. After a considerable number of experimental measurements, the most accurate flop time to use considering theimplemented code and this research machine is 5.7×10^{-6} seconds. Using

this value gives reasonable approximations to the table values in the modeled equations, thus allowing extrapolation of the timing data to higher dimensioned cubes.

The improvement in computing times seen in this and the previous chapters are well within the goals established for this research. However, it has not yet been seen as to how well the estimator performs in a variety of DOA estimation situations. The next chapter is provided to demonstrate the performance of the two-eigenvector estimator function in other than speedup considerations.

TABLE 4

OVERALL DIRECTION-OF-ARRIVAL DATA

32 SAMPLES FROM EACH ANTENNA

Number of Nodes	Number of Antennas 16 CTU	S_ρ	64 CTU	S_ρ	96 CTU	S_ρ	160 CTU	S_ρ
1	2.355	1.00	12.42	1.00	21.54	1.00	46.04	1.00
2	1.198	1.97	6.31	1.97	10.96	1.96	23.21	1.98
4	0.648	3.63	3.31	3.75	5.73	3.76	12.17	3.78
8	0.363	6.49	1.78	6.98	3.12	6.90	6.64	6.92
16	0.225	10.47	1.07	11.61	1.88	11.46	4.06	11.34

64 SAMPLES FROM EACH ANTENNA

Number of Nodes	Number of Antennas 16 CTU	S_ρ	64 CTU	S_ρ	96 CTU	S_ρ	160 CTU	S_ρ
1	2.552	1.00	15.42	1.00	28.25	1.00	64.32	1.00
2	1.304	1.96	7.81	1.97	14.31	1.97	32.50	1.98
4	0.706	3.61	4.05	3.81	7.41	3.81	16.81	3.83
8	0.387	6.59	2.15	7.17	3.95	7.15	8.98	7.16
16	0.238	10.72	1.27	12.14	2.30	12.28	5.23	12.30

160 SAMPLES FROM EACH ANTENNA

Number of Nodes	Number of Antennas 16 CTU	S_ρ	64 CTU	S_ρ	96 CTU	S_ρ	160 CTU	S_ρ
1	3.149	1.00	24.39	1.00	48.37	1.00	120.33	1.00
2	1.590	1.98	12.30	1.98	24.37	1.98	60.33	1.99
4	0.845	3.73	6.31	3.86	12.43	3.89	30.75	3.91
8	0.453	6.95	3.28	7.44	6.46	7.49	15.94	7.55
16	0.274	11.49	1.82	13.40	3.55	13.62	8.71	13.81

Note: CTU is in seconds, S_ρ is the speedup ratio

CHAPTER IX

ESTIMATOR EMPIRICAL RESULTS

This chapter presents the experimental data resulting from application of the developed theoretical procedures. All computer simulations were completed on the Intel Hypercube iPSC/2 parallel processor using 16 nodes of the 32 that are available on the research machine. Some experimental work was done in the 32 node configuration, however the data is not included in the results because of hardware constraints and some system modifications that did not allow appropriate testing throughout the research period. The new procedures described in the chapters of this dissertation were coded in the FORTRAN computer language. The greatly improved timing output from the earlier chapters resulted in new functions that had several changes from traditional estimator functions. Therefore, this chapter provides three empirical methods to prove out the new two-eigenvector DOA estimator's performance.

The first method is to make a comparison against other DOA estimator simulations. Output data was compared to some previously published studies which provided enough information so that experiments using the new estimator could be completed under similar circumstances.

The computer simulation driver for these experiments is a FORTRAN program that allows the required inputs to be provided for each individual experiment. The simulated noise samples added to the simulated source signals were derived from a group of independent Gaussian noise generators.

The second method within this chapter are several experiments based on real radio data. The data was extracted from the Sampled Aperture Receiving Array (SARA) system which operated a high frequency transmitter located in San Antonio, Texas communicating to a crossed array antenna in Ottawa, Canada (Martin, 1988). The real radio results were provided to verify the performance of the computer simulation procedures as well as the new estimator itself. Both of these items are addressed in this set of experiments.

Finally, a third set of experiments were run to demonstrate the accuracy of the estimator under varying conditions independent of any other studies. They show in a controlled environment, the two-vector estimator's performance.

Comparisons to Published Results

A linear array was used in all cases for this DOA estimator. In some cases, the simulator for this system could not identically repeat the reference experiment because of an excessive number of samples, unique antenna geometries, varied or unknown correlation between sources, etc. In these situations, the reference experiment's setup is indicated by being placed in parenthesis and the values used in the new experiments are stated first, without parenthesis. When nothing is in parenthesis, the same setup was possible.

The most frequent variation was due to the fact that the simulator's incoherent signal outputs actually provide data only for each source individually, rather than all sources at every sample. Additional antennas, placed within the reference's original aperture are used, thus keeping the total samples taken from each source the same between experiments.

Each experiment will be preceded with a listing and a discussion to detail the particular experiment set up, provide reference and page number,

generally state the previous results as provided by the original author, and then briefly describe the results of this experiment. A figure of the function amplitude output for each experiment follows the discussion to graphically reproduce the results obtained in this research. The actual numerical estimates of the DOAs estimated are also provided, as these values are the actual system outputs . The resolution of the DOA in all of these experiments is one tenth of a degree.

Experiment Number 1

DOA	-44 degrees at 25 dB, 30 degrees at 10 dB
Correlation	0
Number of antennas	6 (3)
Interelement spacing	.2λ(λ/3 to center equilateral triangular array)
Number of samples	100
Output	-44.0, 30.1, Figure 21
Reference	Schmidt, 1981 : page 15

The MUSIC algorithm results revealed that there is little or no bias error and that there was no ambiguity problem in the example.

Similar performance can be seen with the two-vector estimator.

Experiment Number 2

DOA	18 degrees at 30 dB, 22 degrees at 30 dB
Correlation	0
Number of antennas	16 (8)
Interelement spacing	.25λ (.5λ)
Number of samples	100 (10 runs with overlay applied)
Output	18.0, 22.0, Figure 22
Reference	Bronez, 1983 : page 130

The reference method worked well at the high SNR.

The new method also works well at high SNR.

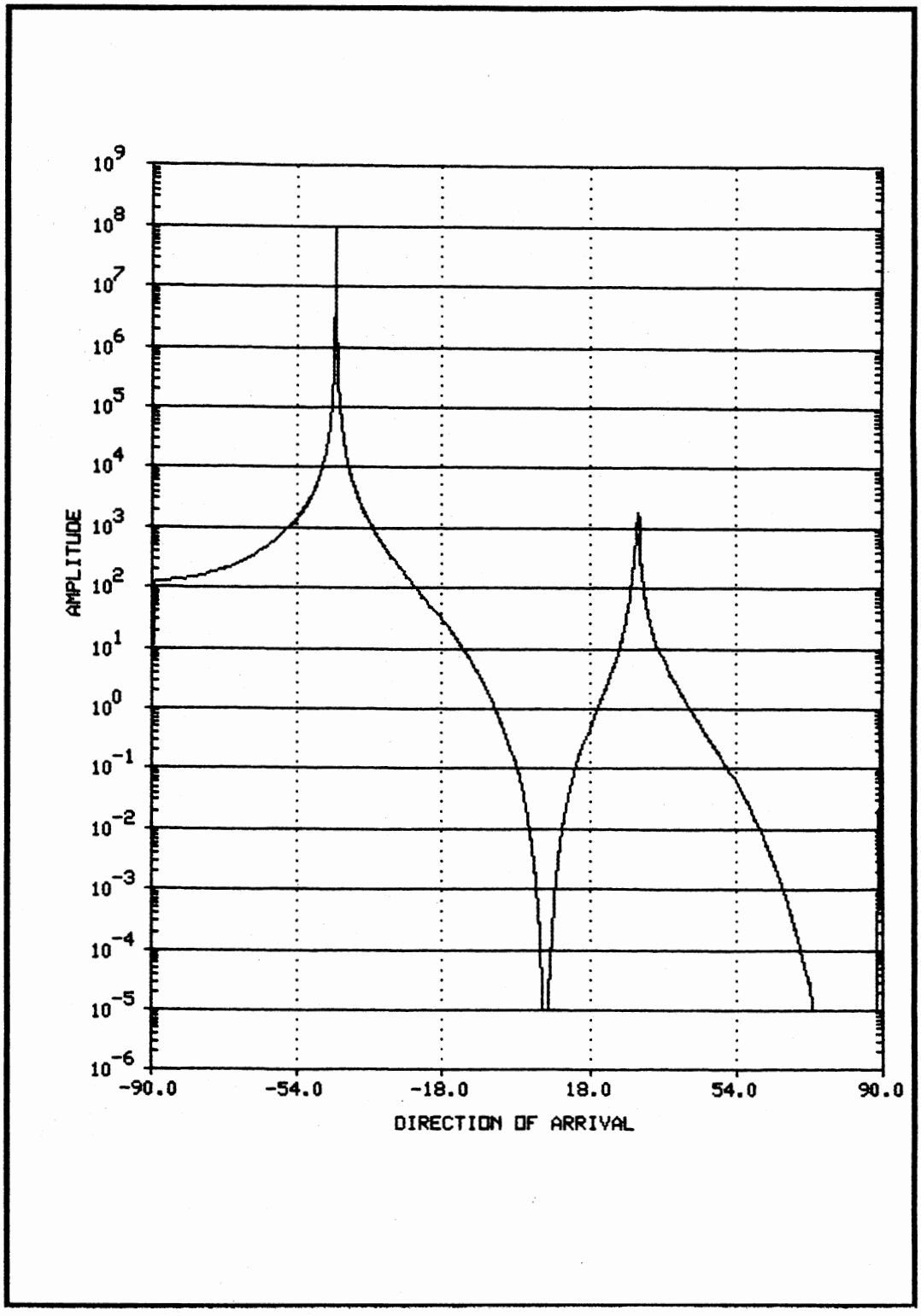


Figure 21. Experiment Number 1, Plot of $\xi(\theta)$ vs DOA Bearing

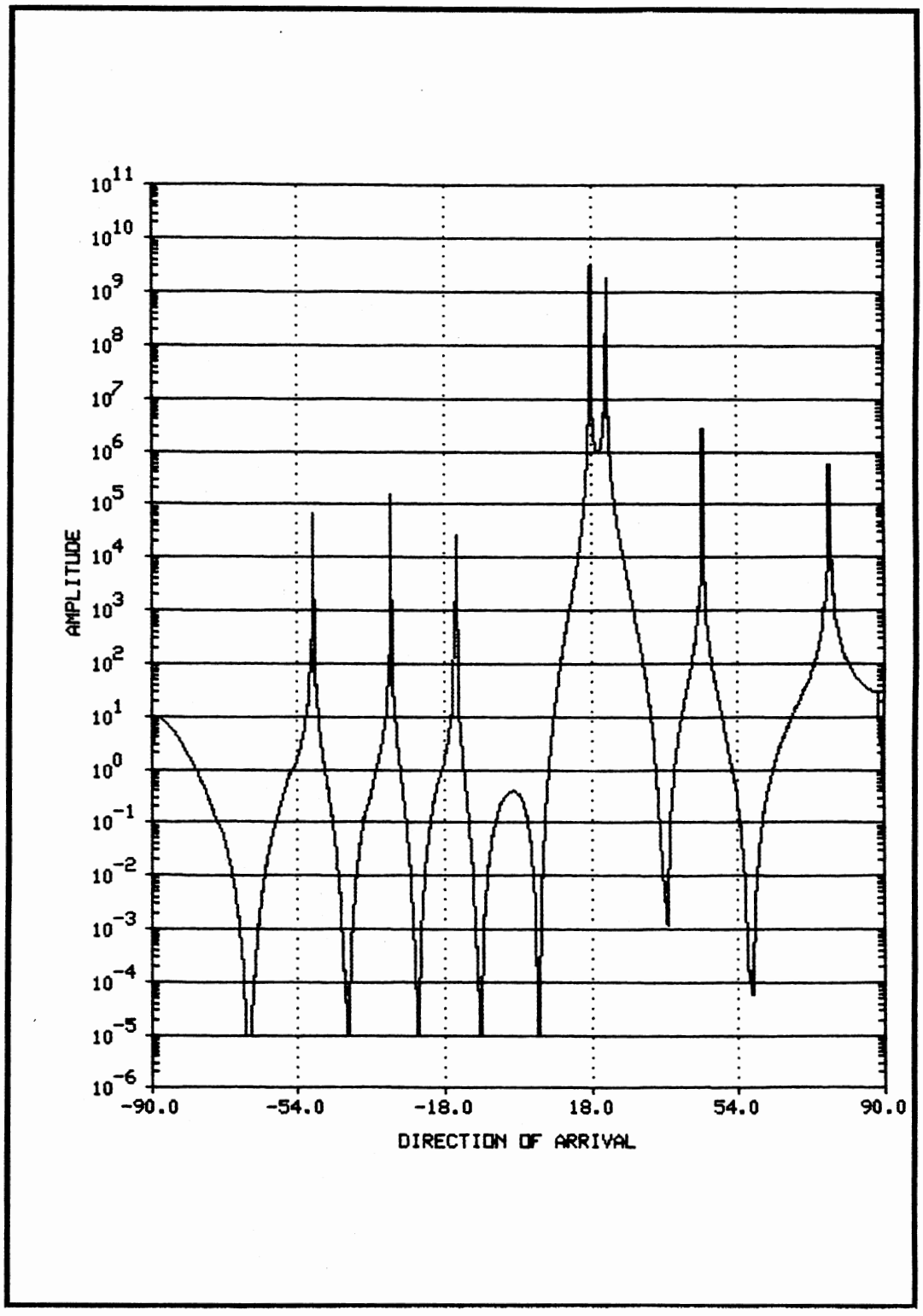


Figure 22. Experiment Number 2, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 3

DOA	18 degrees at 10 dB, 22 degrees at 10 dB
Correlation	0
Number of antennas	8
Interelement spacing	.25λ (.5λ)
Number of samples	50 (25, 10 runs with overlay applied)
Output	18.2, 21.4, Figure 23
Reference	Bronez, 1983 : page 130

The reference method suffered from inaccuracy at low SNR with few snapshots.

The method of this research was more accurate with only slight inaccuracy.

Experiment Number 4

DOA	18 degrees at 10 dB, 22 degrees at 10 dB
Correlation	0
Number of antennas	80 (8)
Interelement spacing	.25λ (.5λ)
Number of samples	100 (500, 10 runs with overlay applied)
Output	17.9, 21.8, Figure 24
Reference	Bronez, 1983 : page 130

The reference method showed resolution between the signals possible even with low SNR, given enough snapshots.

Similar results were obtained but higher more distinct peaks were achieved using only the minimum and maximum eigenvectors.

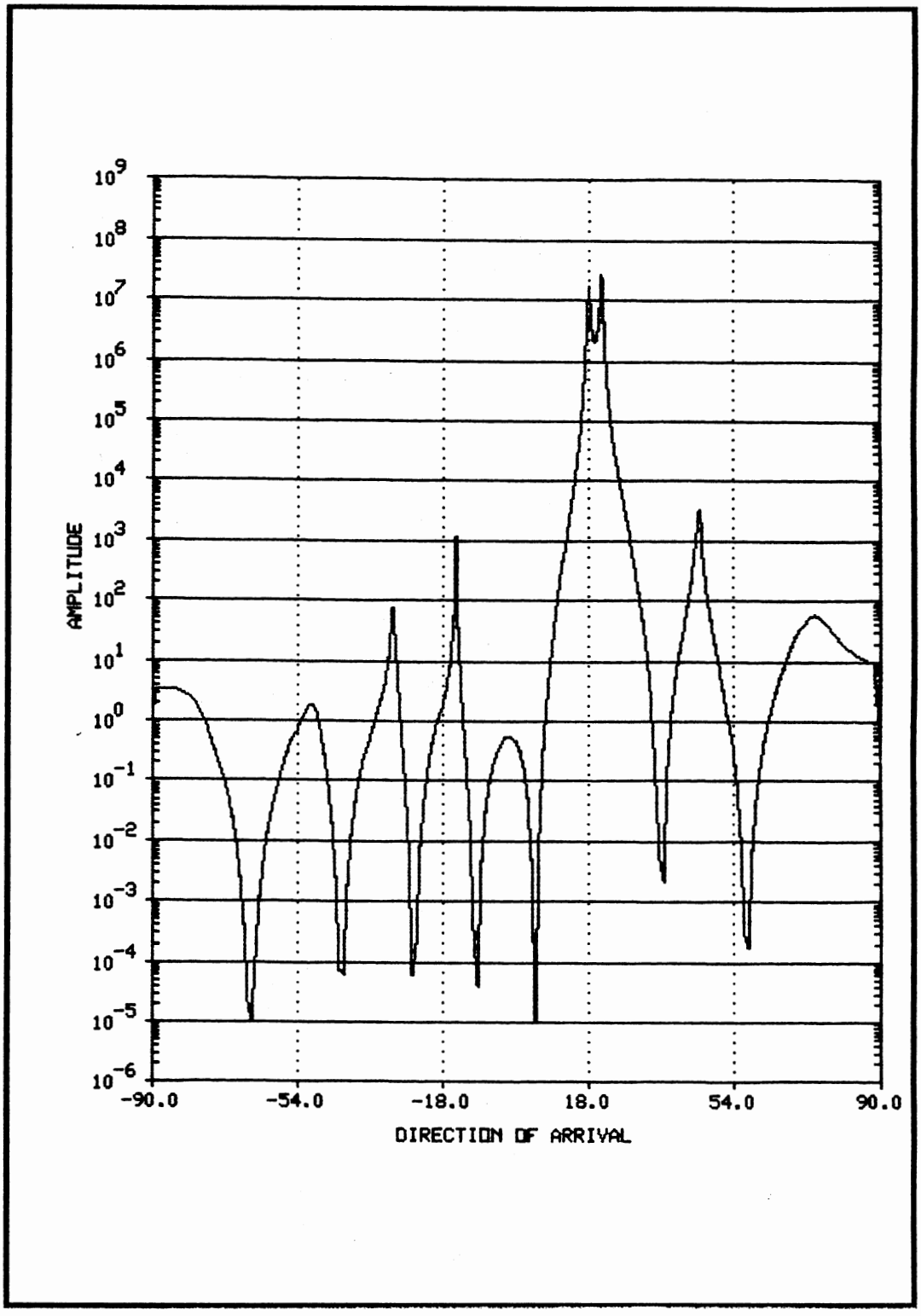


Figure 23. Experiment Number 3, Plot of $\xi(\theta)$ vs DOA Bearing

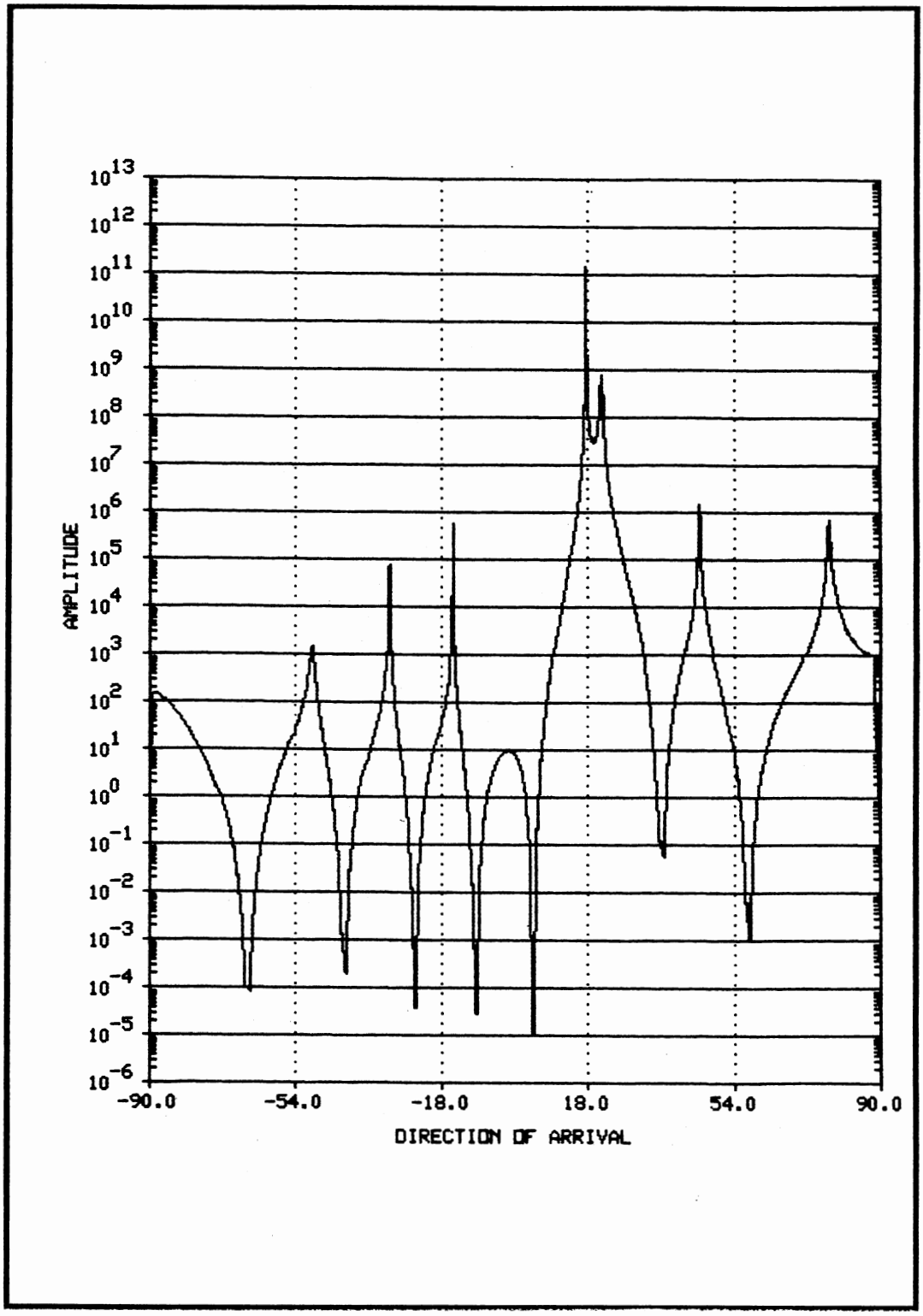


Figure 24. Experiment Number 4, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 5

DOA	18 degrees at 10 dB, 22 degrees at 10 dB
Correlation	0
Number of antennas	8
Interelement spacing	.25λ (.5λ)
Number of samples	50, 7 runs with overlay (25, 10 runs)
Output	average values 18.0, 22.5, Figure 25
Reference	Bronez, 1983 : page 130

The reference method showed resolution between the signals possible.

Again, similar results were obtained but higher more distinct peaks were achieved using the minimum and maximum eigenvectors. This figure has 7 independent runs overlayed to show consistency and the average of the outputs is listed above.

Experiment Number 6

DOA	0 degrees at 0 dB
Correlation	0
Number of antennas	10
Interelement spacing	.5λ
Number of samples	50
Output	0.2, Figure 26
Reference	Johnson, 1982 : page 641

The reference method showed accurate and fine resolution possible even with very low SNR, given enough snapshots.

The minimum and maximum eigenvectors gave similar results. A high peak was achieved, however the maximum eigenvector had all of the information.

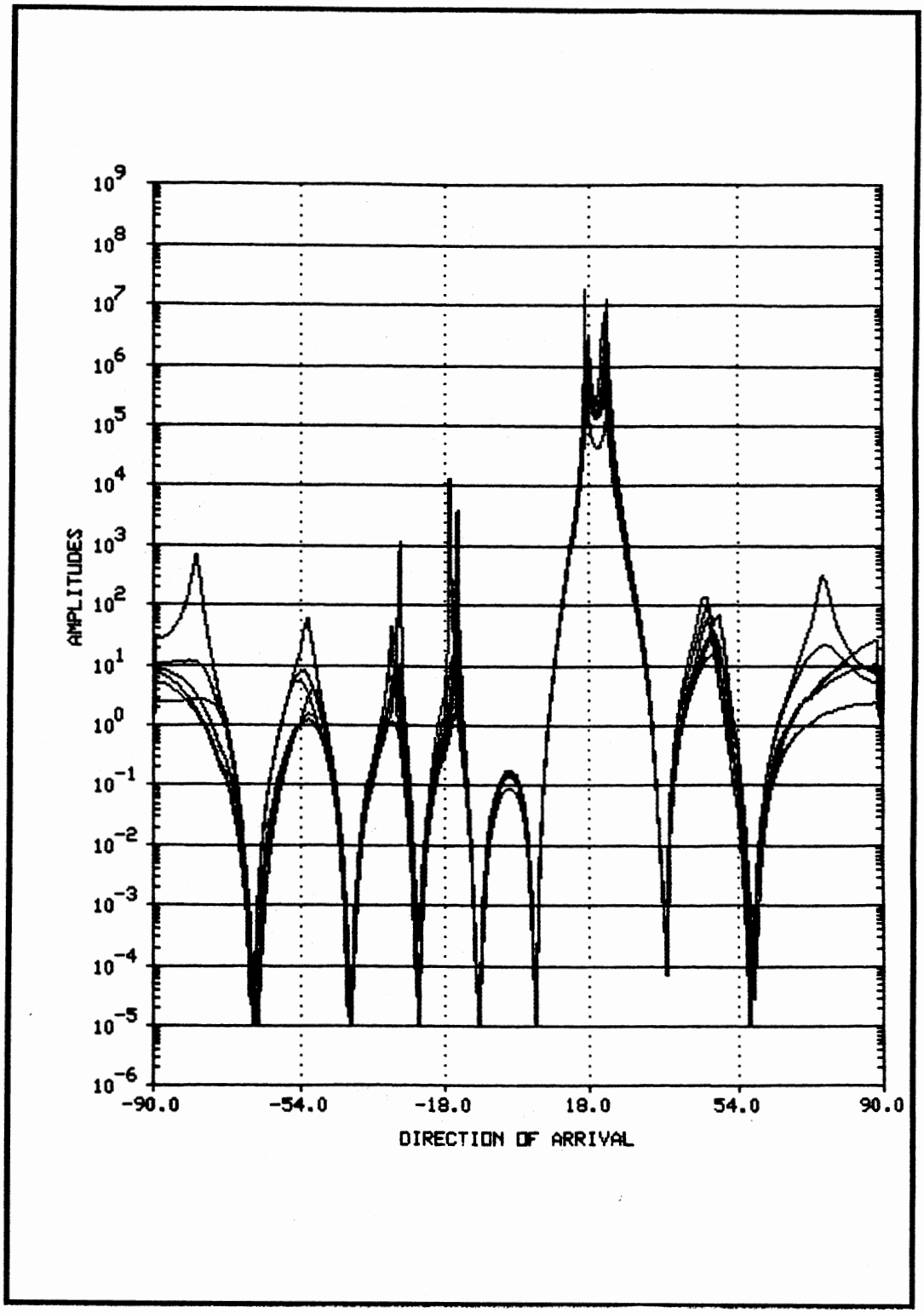


Figure 25. Experiment Number 5, 7 Plot overlay of $\xi(\theta)$ vs DOA Bearing

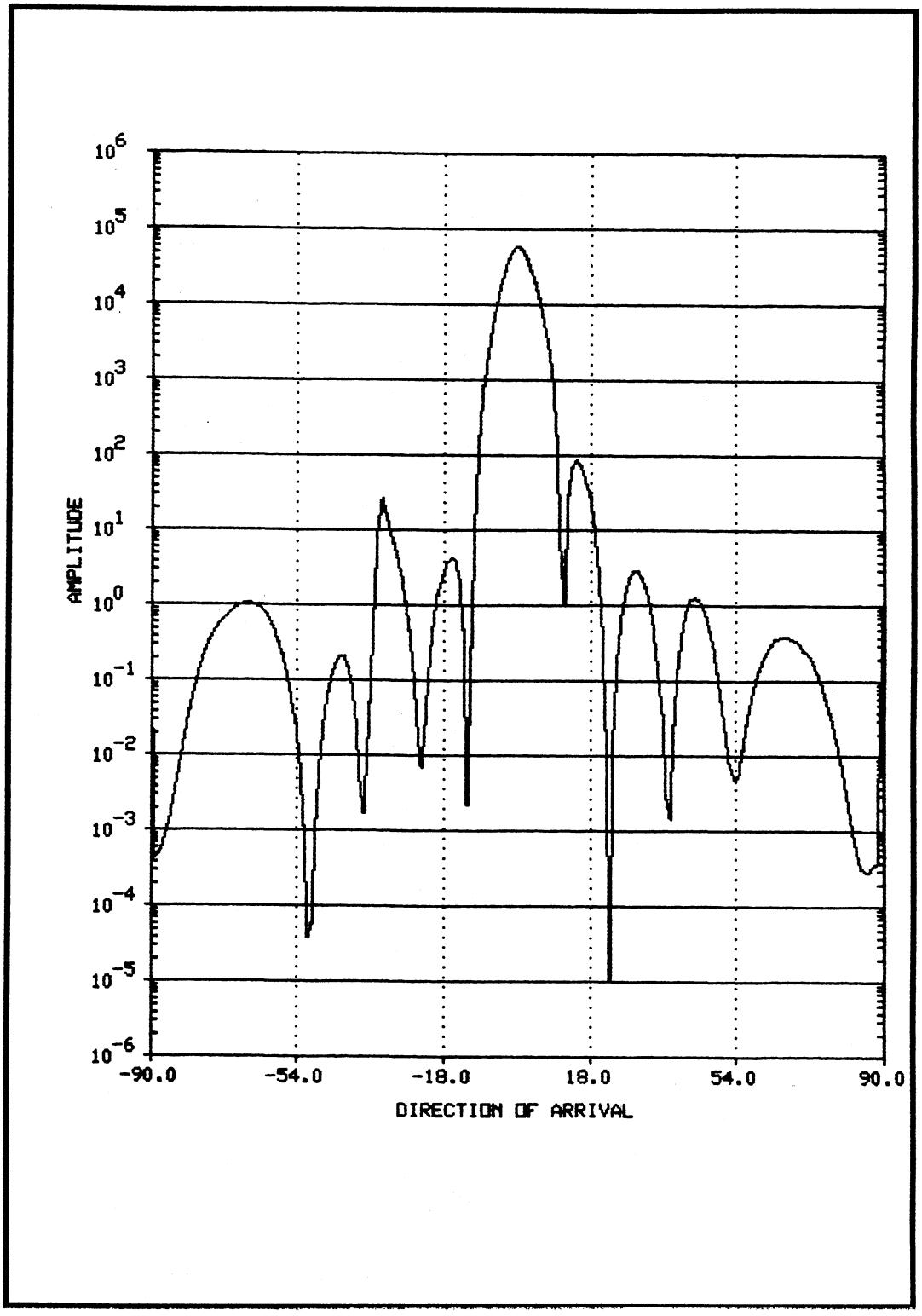


Figure 26. Experiment Number 6, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 7

DOA	0 degrees at 0 dB
Correlation	0
Number of antennas	50 (10)
Interelement spacing	.1λ (.5λ)
Number of samples	100 (500)
Output	2.5, Figure 27
Reference	Johnson, 1982 : page 641

Increased number of samples improved the energy in the peak for the reference method.

The new method provided less accurate results but has a higher peak . The maximum eigenvector has correct DOA information but was slightly biased off by the minimum eigenvector component.

Experiment Number 8

DOA	5 degrees at 0 dB, -5 degrees at 0 dB
Correlation	0
Number of antennas	16 (8)
Interelement spacing	.25λ (.5λ)
Number of samples	100
Output	-5.0, 4.7, Figure 28
Reference	Johnson, 1982 : page 641

The reference method showed excellent accuracy even with very low SNR.

Equal or slightly better results using the dissertation method, with a much higher peak amplitude.

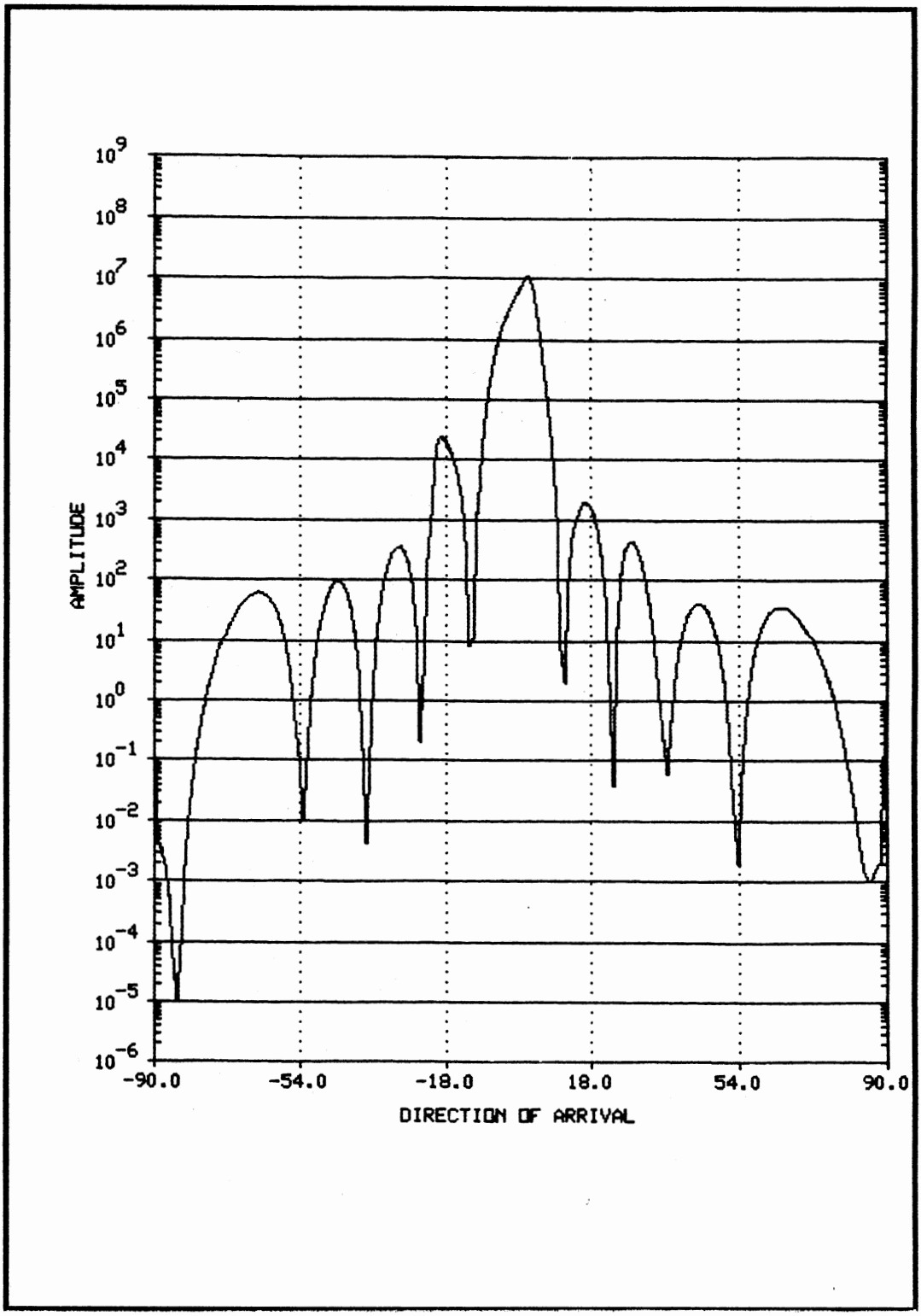


Figure 27. Experiment Number 7, Plot of $\xi(\theta)$ vs DOA Bearing

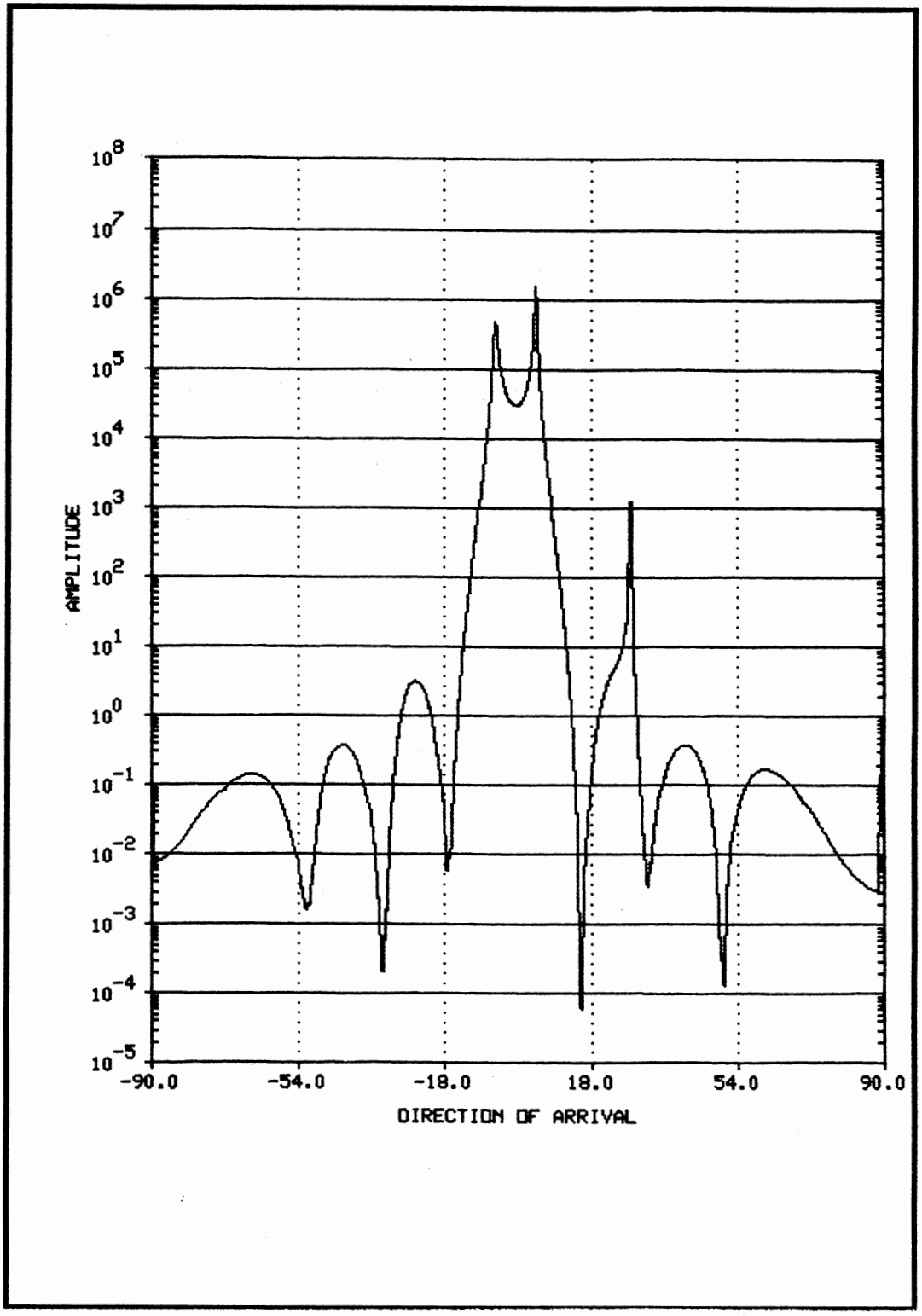


Figure 28. Experiment Number 8, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 9

DOA	3 degrees at 0 dB, -3 degrees at 0 dB
Correlation	0
Number of antennas	16 (8)
Interelement spacing	.25λ (.5λ)
Number of samples	100
Output	-3.0, 2.9, Figure 29
Reference	Johnson, 1982 : page 643

The reference method lowered accuracy bringing angles closer together.

Better results were obtained using the proposed method, with higher peak amplitudes and a deeper amplitude dip separating the two sources.

Experiment Number 10

DOA	-30 degrees at -15 dB, -22 degrees at -18dB, -15 degrees at 15 dB
Correlation	0
Number of antennas	48 (8)
Interelement spacing	.084λ (.5λ)
Number of samples	150 (300)
Output	-27.2,-15.0, Figure 30
Reference	Paulraj, 1986 : page 13

The reference method showed higher accuracy and resolution was distinct .

The two eigenvector method showed the weaker results in this experiment, missing the -22 degree peak completely. The extremely low SNR caused the -30 and -22 degree sources to coalesce into a single wavefront.

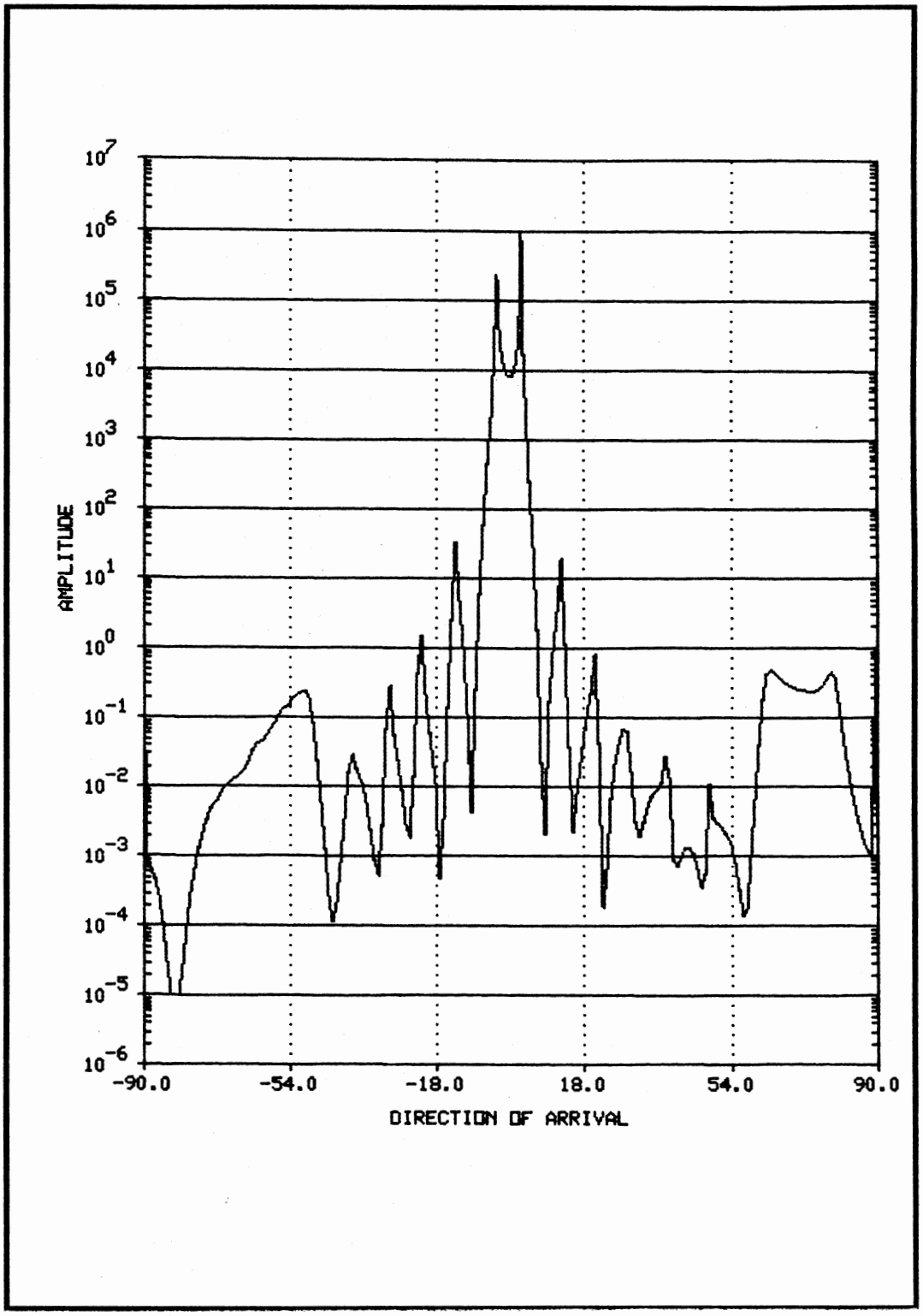


Figure 29. Experiment Number 9, Plot of $\xi(\theta)$ vs DOA Bearing

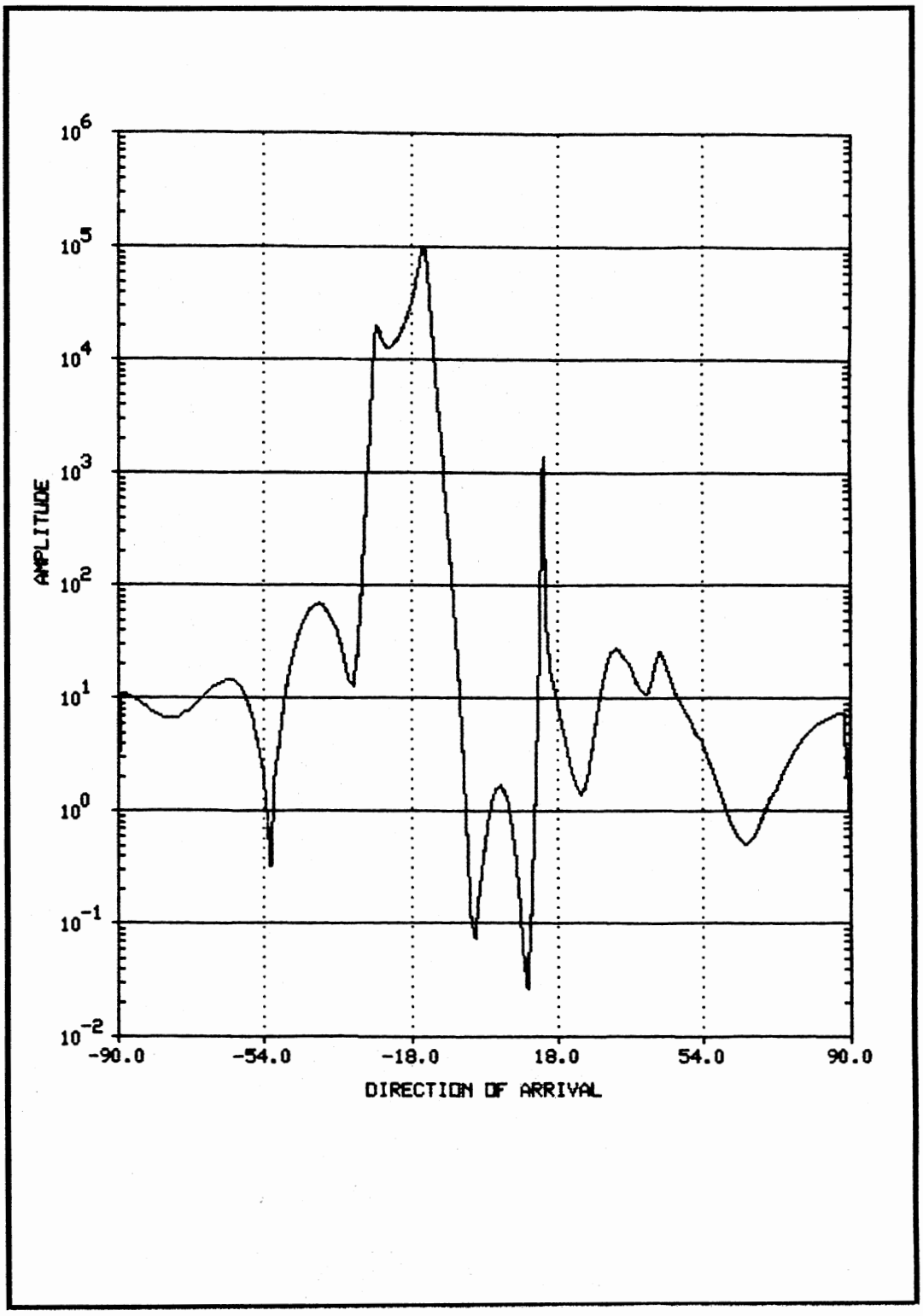


Figure 30. Experiment Number 10, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 11

DOA	-60 degrees at 5 dB, -5 degrees at 6 dB, 20 degrees at 4 dB, 45 degrees at 2 dB
Correlation	0 , .6 (stated to be correlated)
Number of antennas	48 (6)
Interelement spacing	0.0375λ (.5λ)
Number of samples	160 (500)
Output	-59.8, -5.1, 20.1, 45.1, Figure 31, -53.7, -8.5, 19.7, 49.6, Figure 32
Reference	Williams, 1986 : page 429

The reference method showed very high accuracy and resolution was very distinct. This is an improved spatial smoothing technique, which works well with coherent signals.

The four wavefronts are also identified using the two vector procedure however there is some inaccuracy. The low SNRs used in the experiment a problem, however using the highly correlated signals caused most of the inaccuracy in this procedure. The results were as good as or better as the reference's MUSIC and conventional spatially smoothed methods. Two separate runs were accomplished. The first using incoherent sources, Figure 31, and the second using a .6 correlated pair of sources Figure 32.

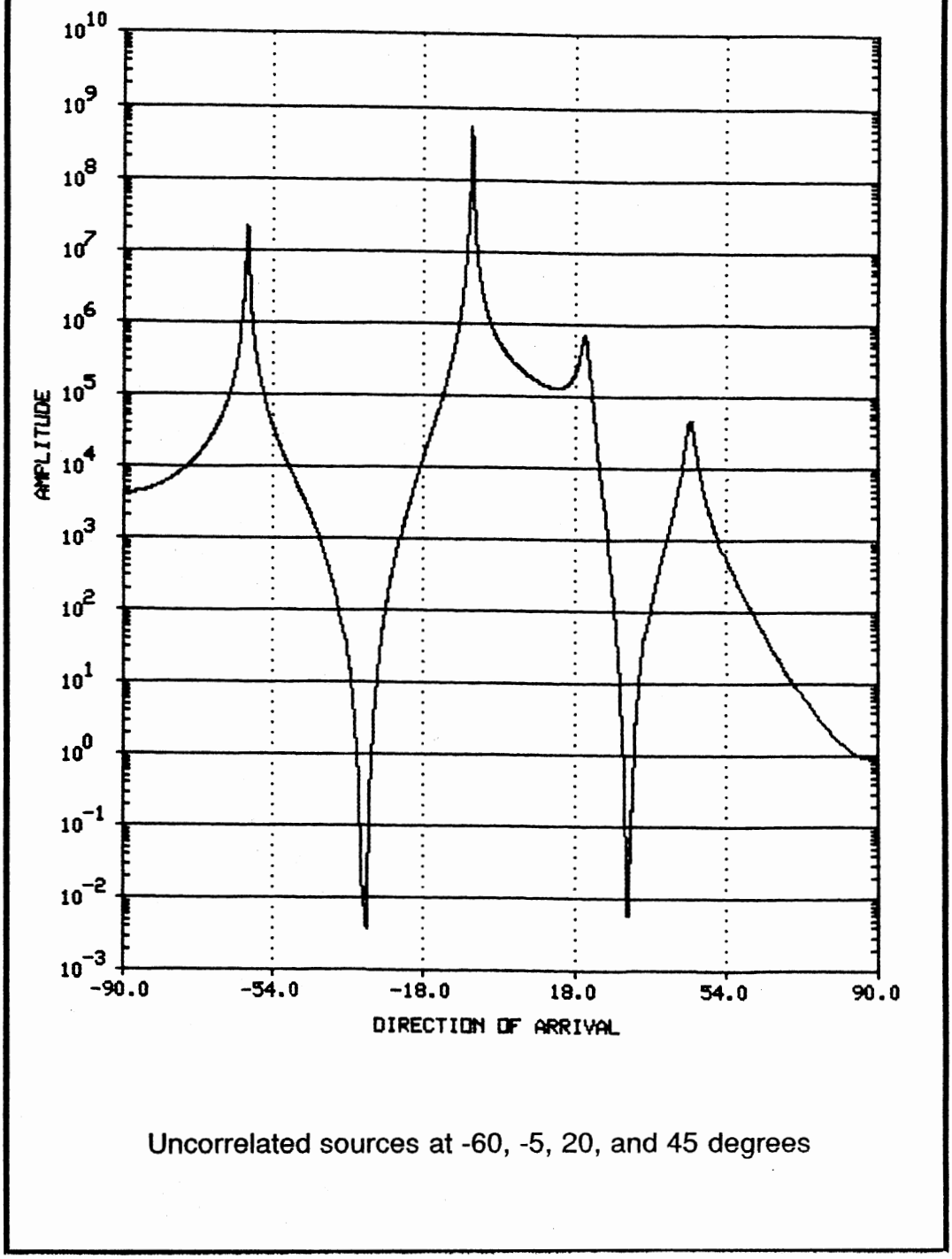


Figure 31. Experiment Number 11, Plot of $\xi(\theta)$ vs DOA Bearing

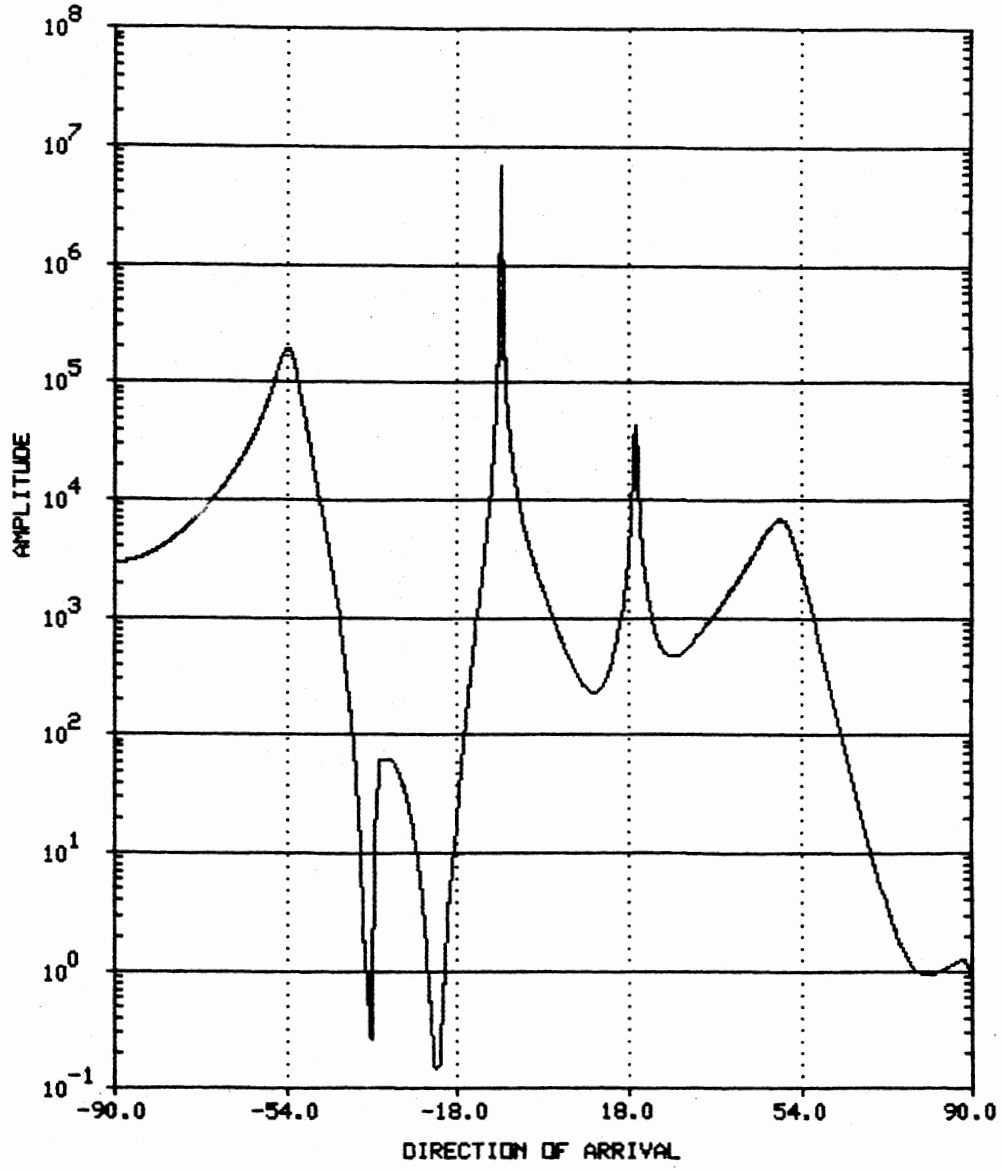


Figure 32. Experiment Number 11, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 12

DOA	20 degrees at 20 dB, 30 degrees at 20 dB
Correlation	1
Number of antennas	20 (10)
Interelement spacing	.25λ (.5λ)
Number of samples	100
Output	26.0,34.0, Figure 33
Reference	Williams, 1988 : page 429

Once again high accuracy was reported for the reference method.

The two eigenvector method showed good results. The strong SNR used in the experiment kept the two sources from coalescing into a single wave even when simulated to be totally coherent, but some inaccuracy resulted.

Experiment Number 13

DOA	20 degrees at 20 dB, 30 degrees at 20 dB
Correlation	0 (1)
Number of antennas	20 (10)
Interelement spacing	.25λ (.5λ)
Number of samples	100
Output	20.0, 30.0, Figure 34
Reference	Williams, 1988 : page 429

This is the same as experiment 12, except this simulation used an uncorrelated pair of signals.

The two-vector method showed excellent results. The strong SNR allows the two waves to form very accurate, distinct peaks at high energy levels.

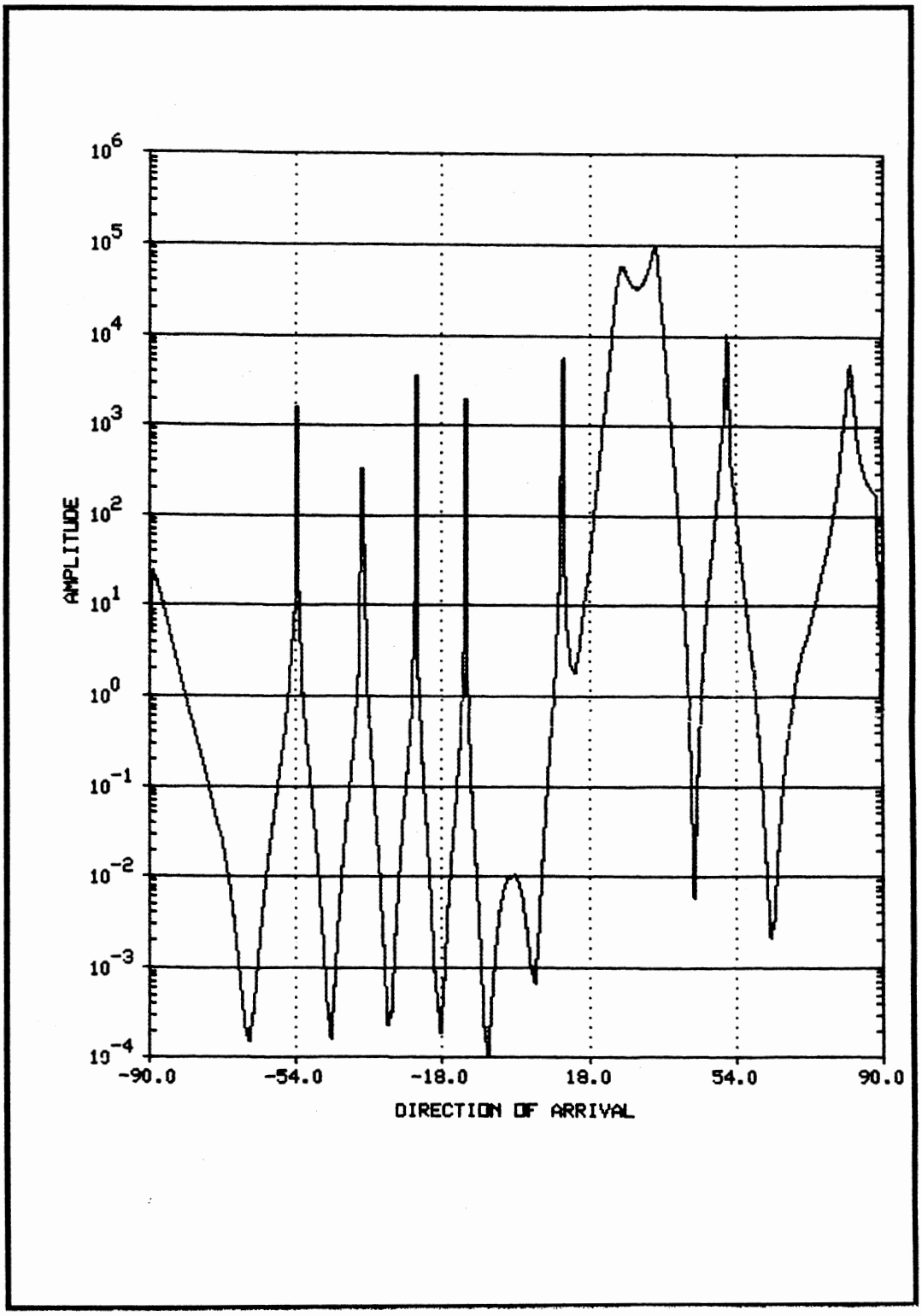


Figure 33. Experiment Number 12, Plot of $\xi(\theta)$ vs DOA Bearing

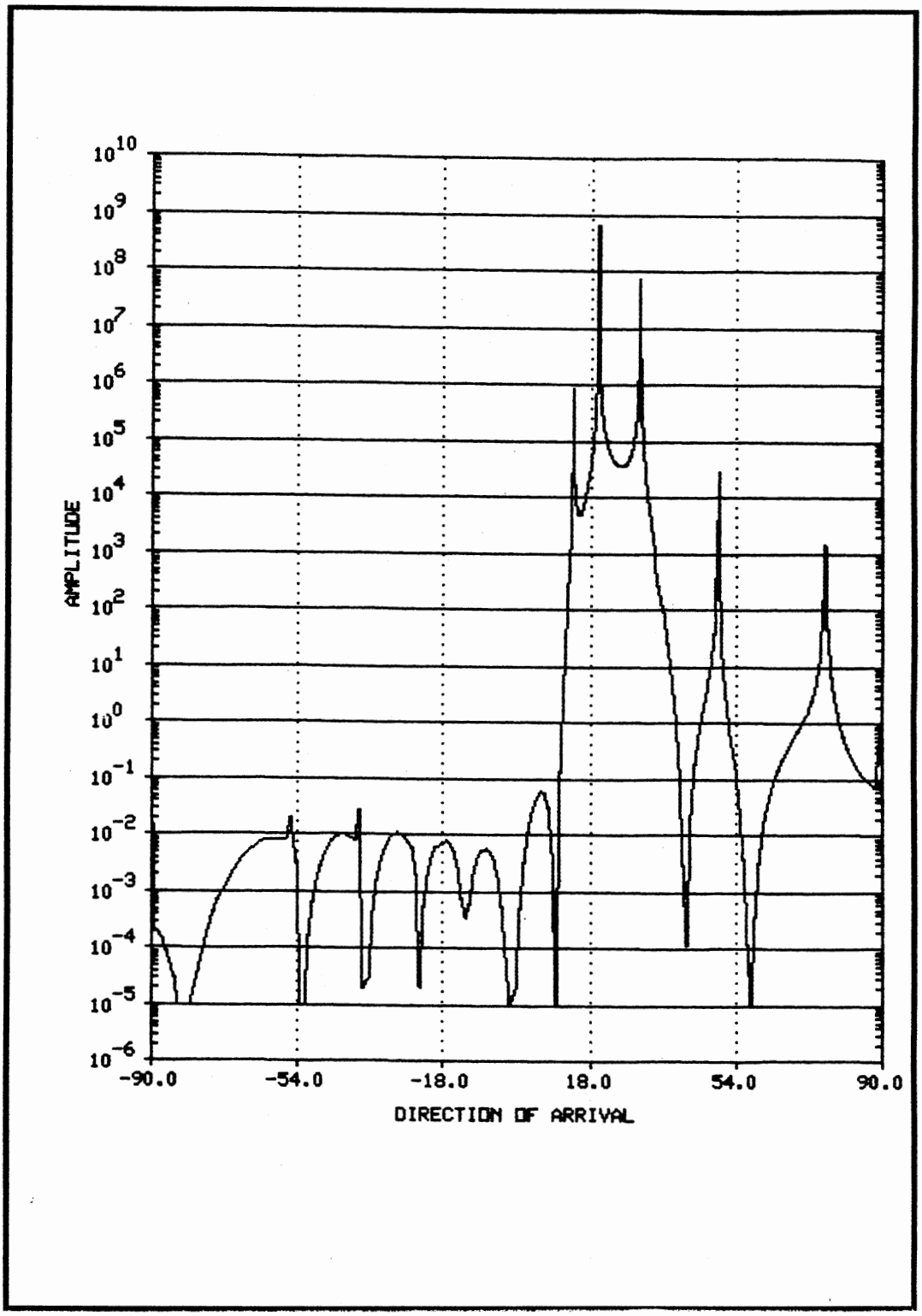


Figure 34. Experiment Number 13, Plot of $\xi(\theta)$ vs DOA Bearing

Results Using Actual Radio Data

Following this discussion are figures which are the results obtained by using real radio data extracted from the Sampled Aperture Receiving Array (SARA) system. This data was from a 1977 DOA experimental session on an antenna array that operated between San Antonio, Texas and Ottawa, Canada. The geometry of the antenna array is a Mills Cross Configuration with a total of 62 antenna elements. The sampled data used here was for only 16 of these elements. The physical arrangement of the "experinental array" was eight vertical monopole antenna elements in a horizontal plane running almost NW-SE, and eight similar elements running nearly NE-SW. A total of 155 samples were available from each antenna element. This set of data has been used by others including (Alsup, 1984, Kaplan, 1987, and Martin, 1988).

Figure 35 is a skeletal drawing of the situation with applicable dimensions and a repeat of the geometric and mathematical relationships from Figure 2. For convenience, the arms are labeled in accordance with their directional orientation with the individual antenna elements numbered from 1 to 16.

A difference between this study's model and the actual array existed in that the arms' element separations were not totally equally spaced. The closest center two elements of both arms have a spacing, 53.34021 meters, which is different than all of the outer elements, 30.48012 meters. In order to work around this problem, colinear subarrays had to be considered in the analysis.

The correct value for azimuth between Ottawa and San Antonio is 6.70 degrees West off the endfire of the SW arm as depicted in the figure. Because of multi-hop propagation nature of the HF band, two possible elevation angles were estimated to be present. The lower DOA elevation angle was estimated to be a single-hop arrival at 3.46 degrees. The next possible elevation angle was

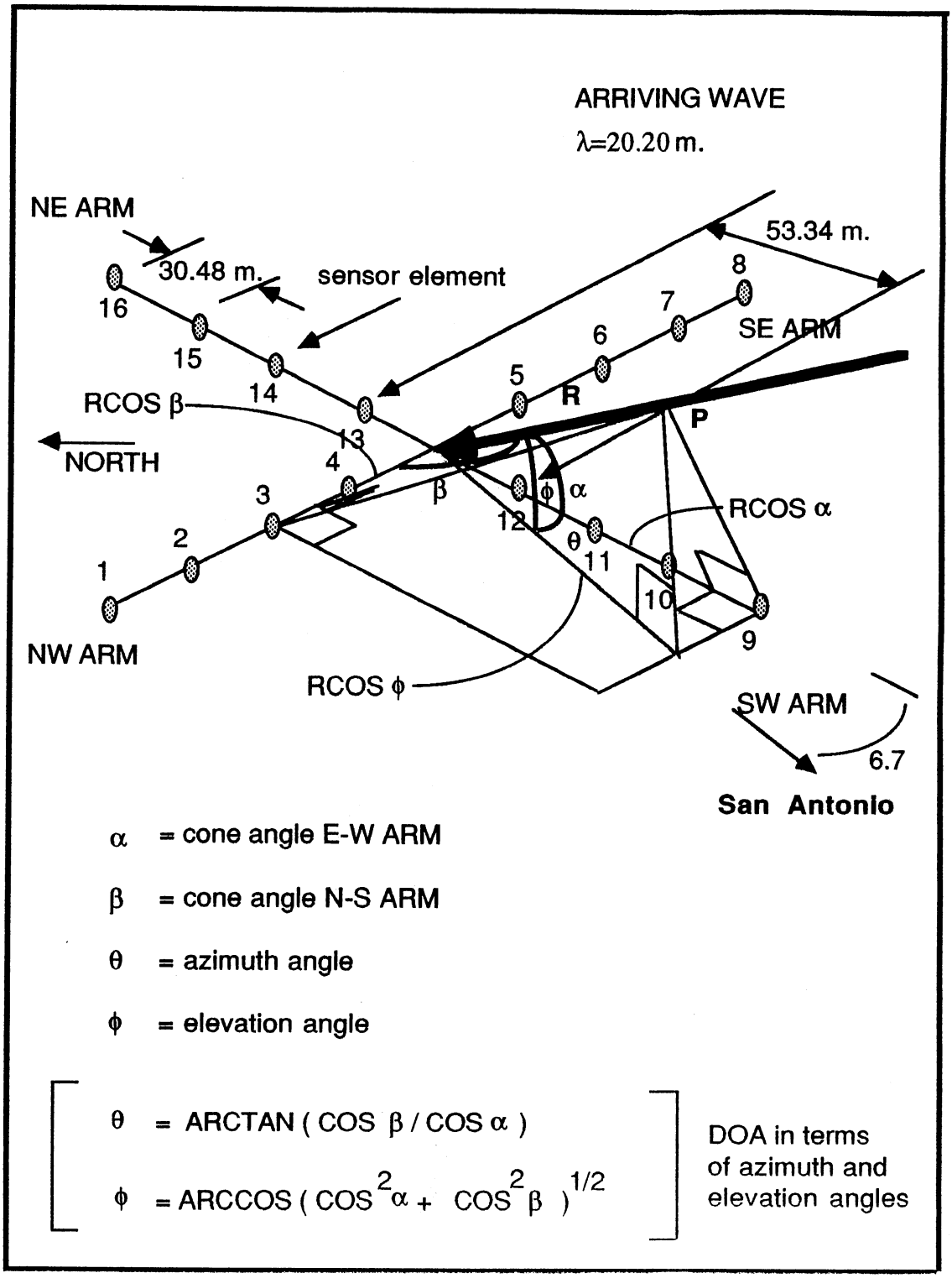


Figure 35. Determination of Azimuth and Elevation For SARA Radio Data

estimated for the two-hop situation to be at 15.43 degrees. Dominant marrivals in the experiment were determined to be the one-hop and two-hop F2 modes.

The SARA data was used to validate the simulation procedures, and of course, also validate the two-vector estimator performance. It should be noted that the SARA data was previously determined to not actually be error free due to measurement or calibration considerations (Alsup, 1984, Kaplan, 1987, and Martin, 1988). Efforts to resolve the phase errors through "grooming" of the data usually resulted in a phase multiplier applied to the last eight antenna outputs in the above studies. It was also determined that since the multi-hop receive situation was occurring, subsets of the samples were grouped, or isolated, to allow only the single-hop arrival data to affect the analysis.

None of these correcting efforts were applied to the SARA radio data in this DOA estimation attempt. Rather, in this case, the approach was to simply take all of the samples for each of the antennas unaltered, as if no knowledge of these problems existed, and analyse the output.

As mentioned above, the nonlinear nature of the array, the different spacing for the center elements, did not allow all eight elements to be processed concurrently as a colinear array. Instead, it was required to split each arm of the array into three segments, the two outer arms using four antenna elements each, and then the inner segment using the two center antenna elements.

After each of the segment results were obtained, the multiple cone angle estimates were averaged to arrive at the single result for each cone angle. Then these two angles were used to compute the elevation and azimuth angles, or a three dimensional DOA.

Figures 36 through 41 are six plots that resulted from the data processed, and are labeled according to the applicable arm segment of the antenna array.

The estimate arrived at for the azimuth using the average of the values

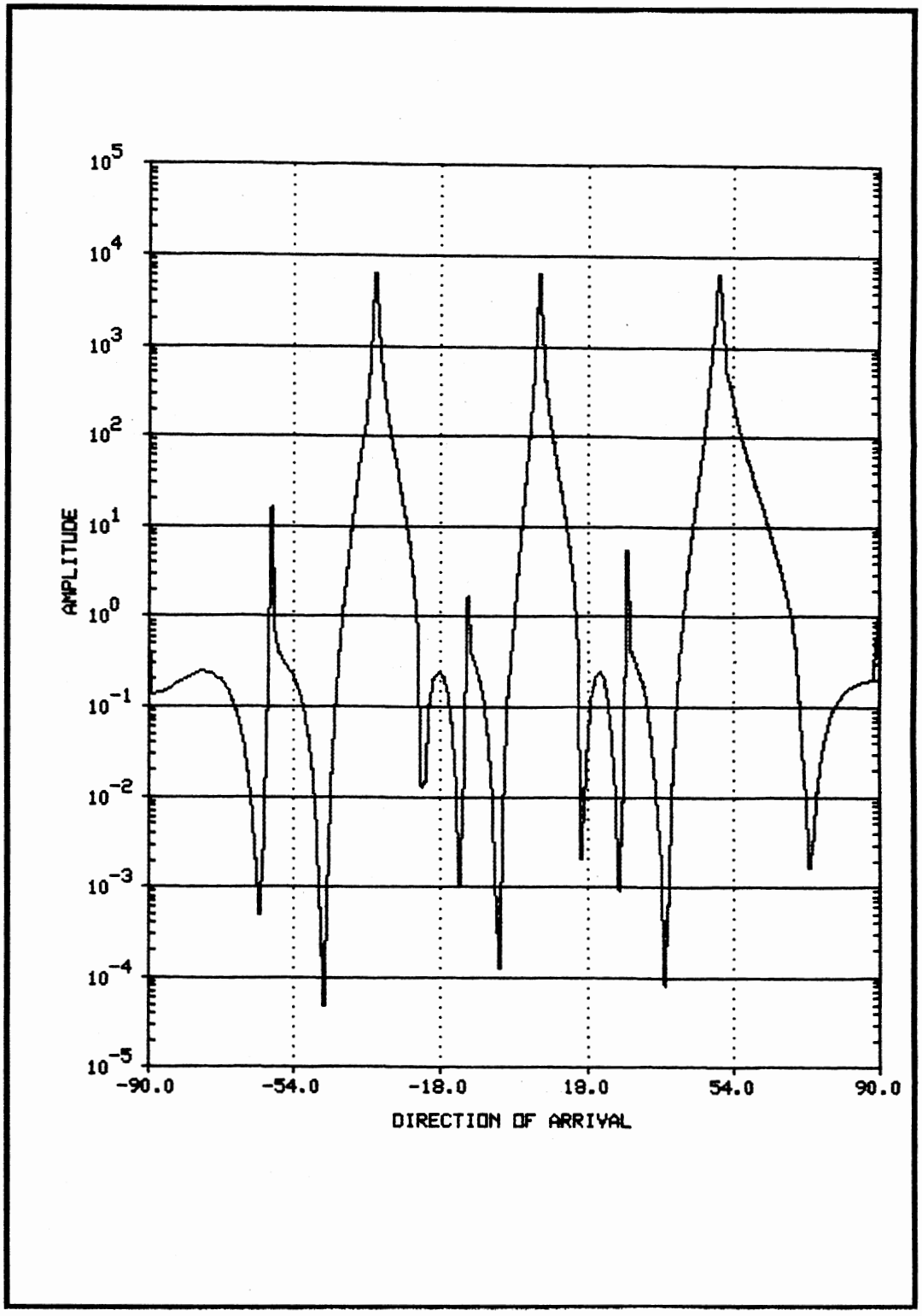


Figure 36. SARA Data, Plot of $\xi(\theta)$ vs DOA Bearing for NW Arm

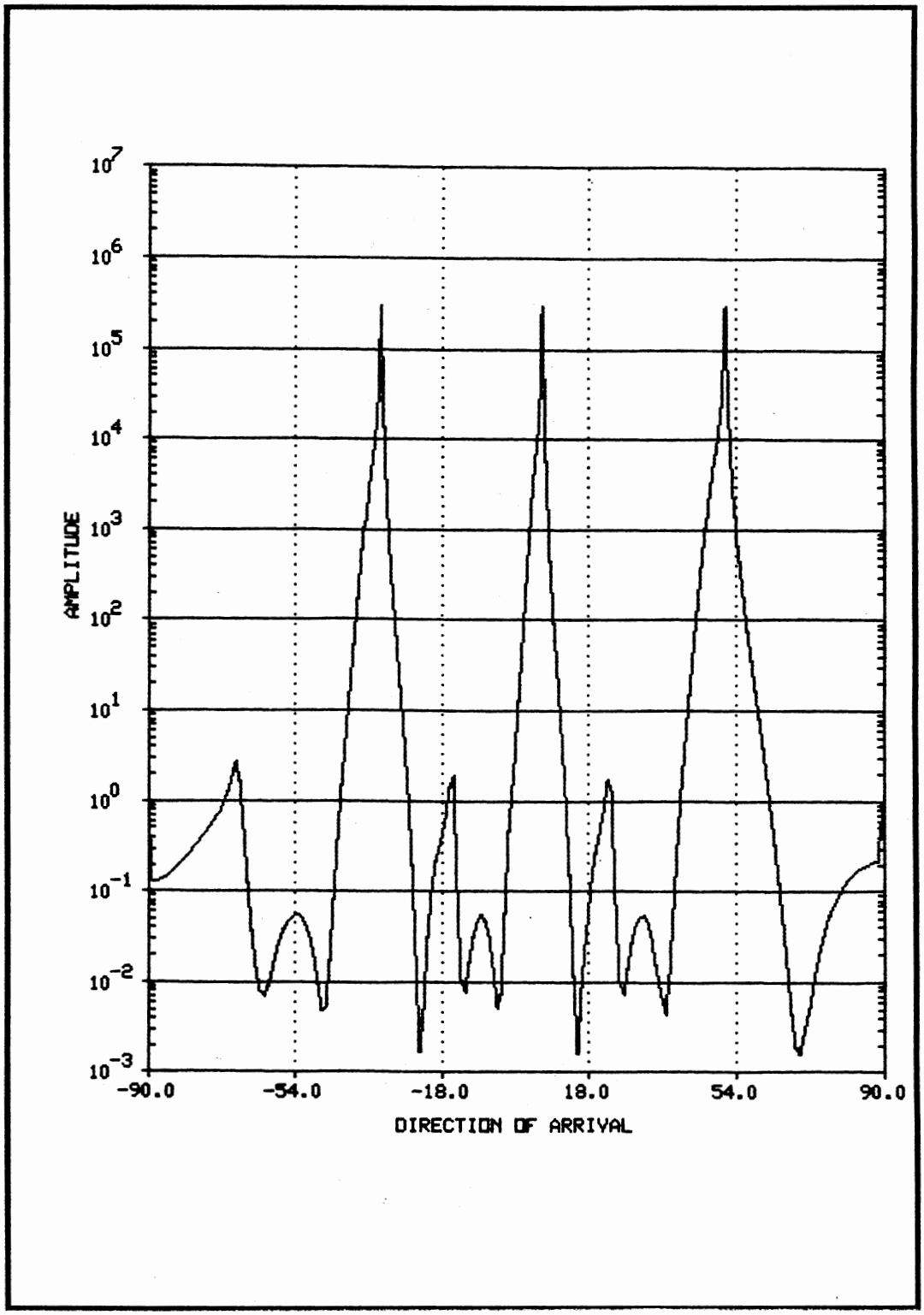


Figure 37. SARA Data, Plot of $\xi(\theta)$ vs DOA Bearing for SE Arm

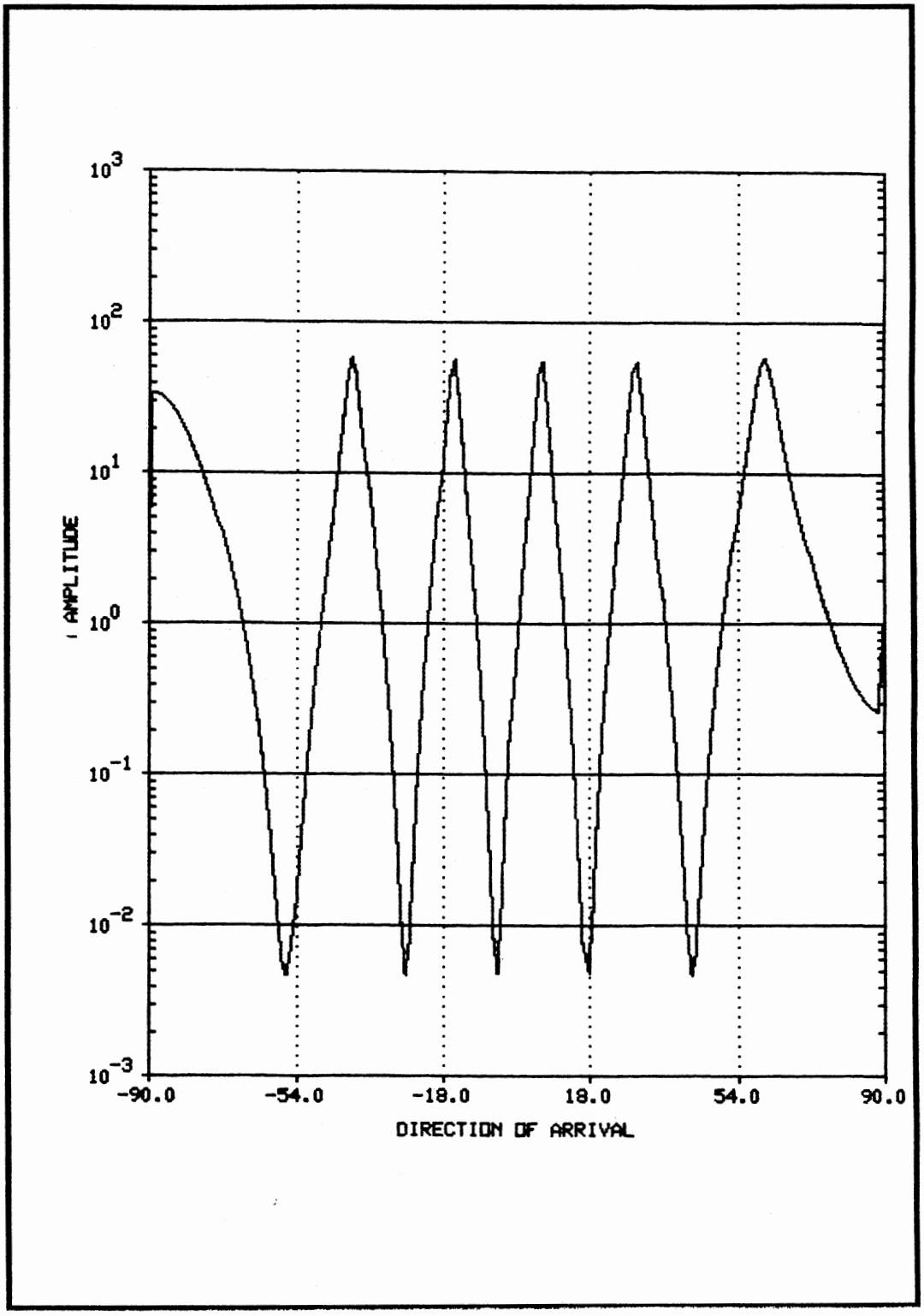


Figure 38. SARA Data, Plot for NW/SE Arm, Center Two Elements

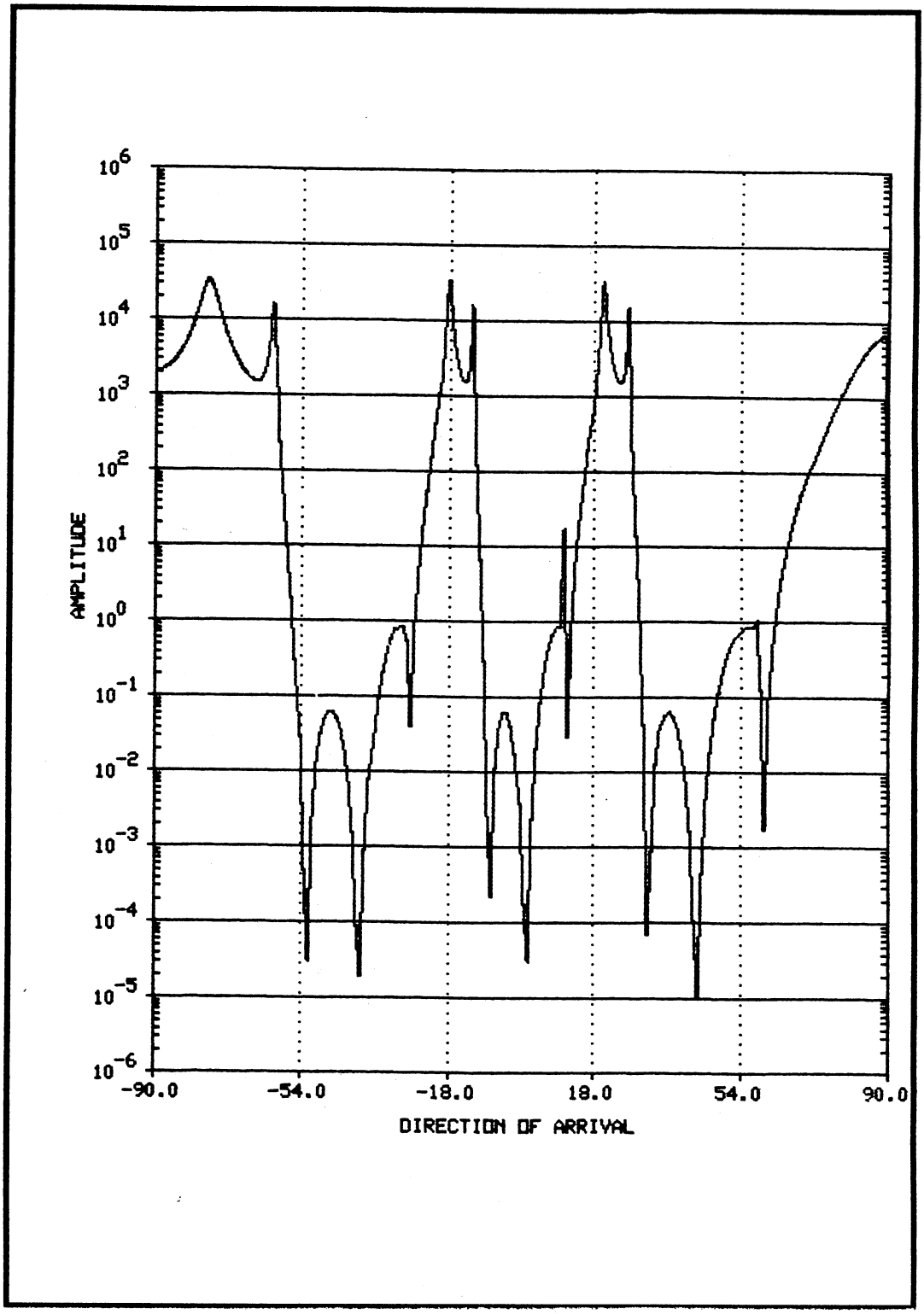


Figure 39. SARA Data, Plot of $\xi(\theta)$ vs DOA Bearing for SW Arm

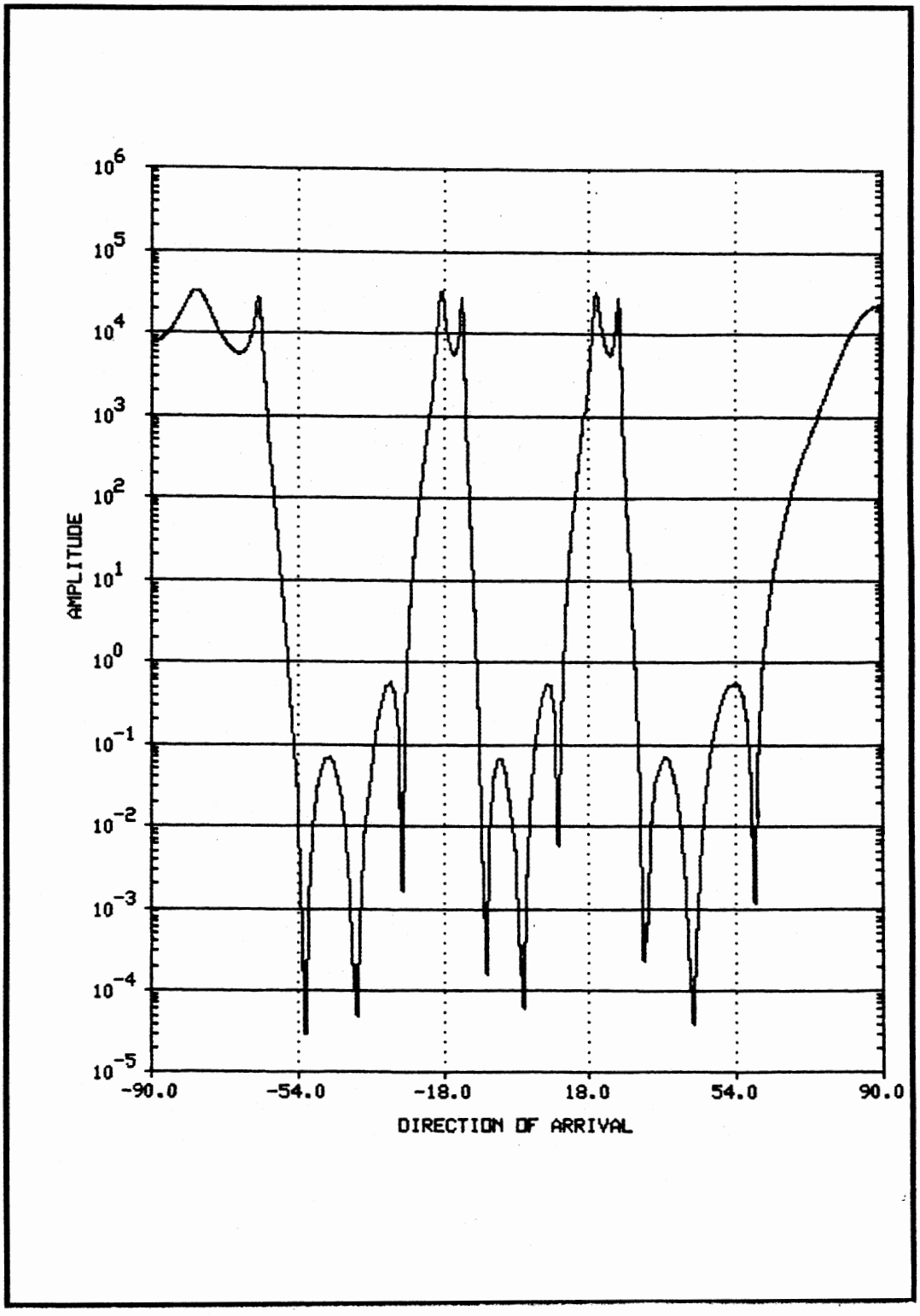


Figure 40. SARA Data, Plot of $\xi(\theta)$ vs DOA Bearing for NE Arm

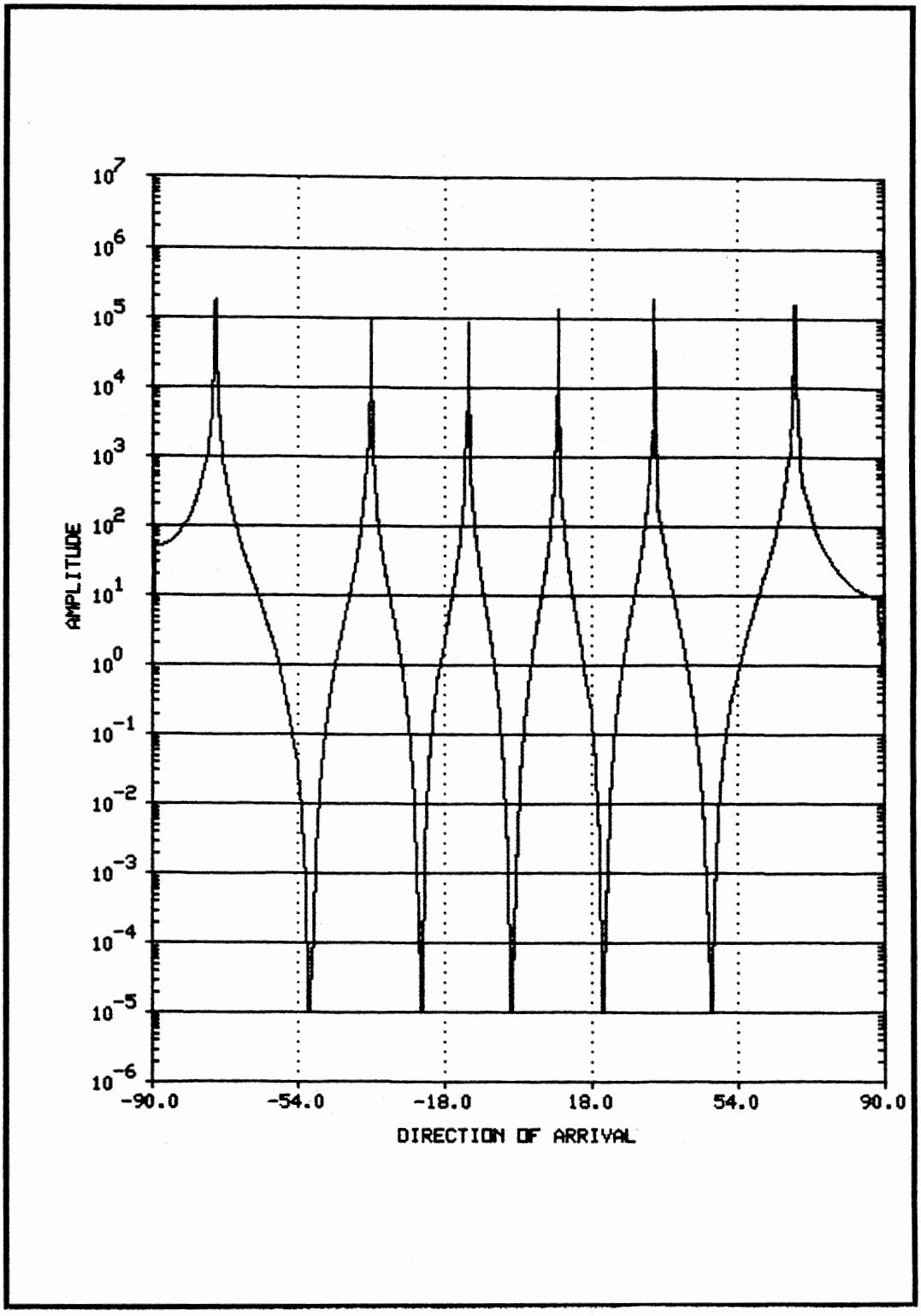


Figure 41. SARA Data, Plot for NE/SW Arm, Center Two Elements

obtained is 6.77 degrees. This is very close to the correct value of 6.7 degrees and is considered close enough to be a reasonable validation for the estimator.

The estimate for the elevation angle using the average of the values was 11.63 degrees. This value is near the two hop value estimate of 15.43 degrees, but obviously this was not as close as the first case. This inaccuracy points out that either some problem with the estimator or the data must exist.

To further extend the comparison, and also hopefully improve the elevation accuracy, groupings of two antennas were processed separately as antenna pairs. This was expected to eliminate some calibration problems by localizing the data to compare phase shift only between the nearest neighbor elements. This procedure resulted in seven values of angle of arrivals for each arm of the antenna array. These different angles-of-arrival estimates were averaged to yield a single cone angle value for each arm as was done above. Applying the mathematics of Figure 35 resolved azimuth and elevation from the cone angles.

It turned out that this straight forward averaging procedure did result in an improved elevation estimate. In this case, the new azimuth estimate remained very close at 6.63 degrees, and the elevation estimate improved to become 14.84 degrees, or about 0.6 of a degree off from the estimate provided in the references for the two hop DOA. Without grooming the data, valid indications of the single hop elevation were not obtained. Taking into account the stated calibration problems of the original study, these values of azimuth and elevation are sufficiently close to further validate this research's estimator .

The last consideration with the SARA data is that because the shorter antenna separation was approximately 1.5 times the receive wavelength, aliasing occured. The aliasing can be seen in the figures as extra peaks harmonically related to the actual arriving wavefront. This made it necessary to find some way to rule out the false aliased peaks from the real peaks. A

procedure was followed which allows the extra peaks to be identified by using the nonlinear separation of the center elements. Their greater span is 2.64 wavelengths, instead of the 1.5 wavelength distance of the other elements. The aliased outputs for these antennas occur at different locations when these two pairs of elements were processed as can be seen in Figures 38 and 41. Since the different spacing has no effect on the actual DOA, it is possible to locate the real arriving wave with a comparison of the output data of the two situations. Figures 42 and 43 were generated by combining in an overlay fashion, Figures 36 and 38 and Figures 39 and 41, respectively. This identifies which of the arriving signals are from the original source and which are the false peaks due to the extended antenna separation.

To validate the computer simulator capabilities, two plots were generated by using the parameters of the SARA experiment configuration as the simulated inputs. Figures 44 and 45 are two examples of simulating the inputs used in the real radio experiments, Figures 39 and 41, respectively.

Nearly the same plot including the aliased signals resulted, confirming the computer simulator model accuracy. Since Figure 39, the real data, and Figure 44, the simulated data, use four antennas, the plot differences are greater than they are between Figures 41 and 45 which use only two antennas. This is because the antenna array for the simulator does not include any measurement error between antenna elements, or does not have any calibration problems. Clearly, the more antenna elements, the greater the real data is affected by these kinds of errors and would cause differences in the ideal model.

Considering the differences between the simulation process and the real radio situation and the known difficulties with the real data, it is reasonable to conclude that the computer simulator is reasonably confirmed as an accurate representation of real radio data and noise.

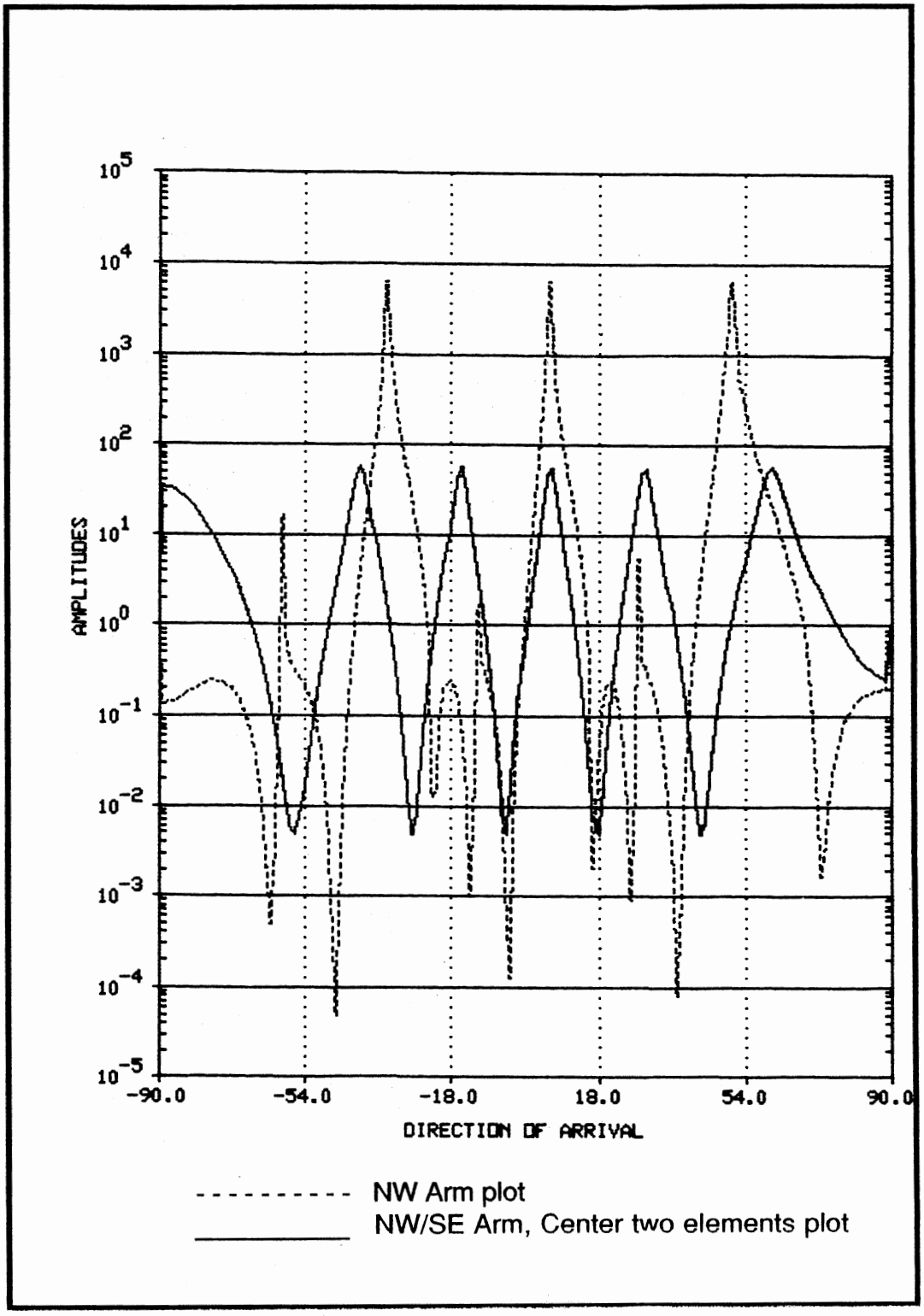


Figure 42. SARA Data, Overlay Plot for NW Arm and NW/SE Arm, Center Two Elements

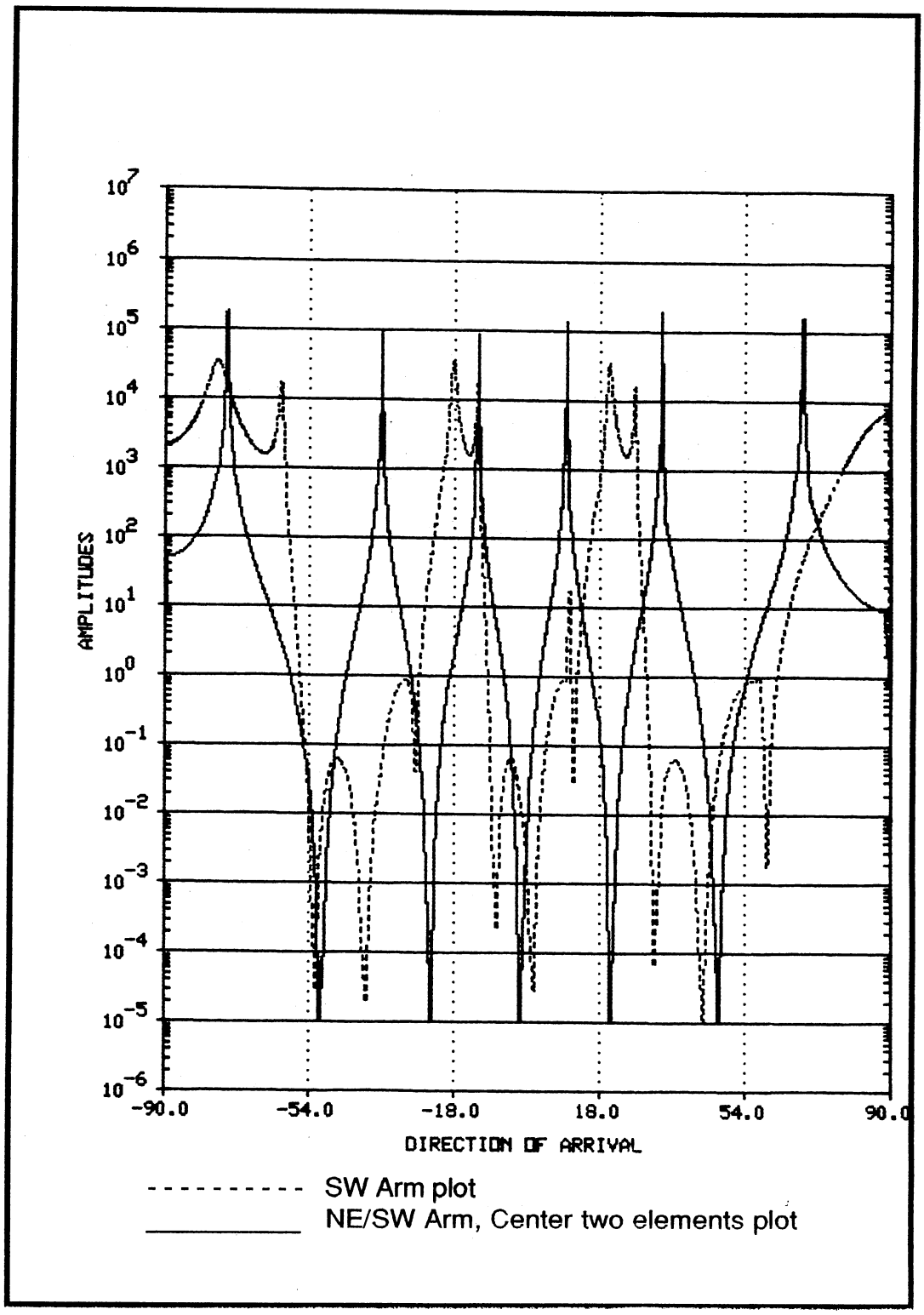


Figure 43. SARA Data, Overlay Plot for SW Arm and NE/SW Arm, Center Two Elements

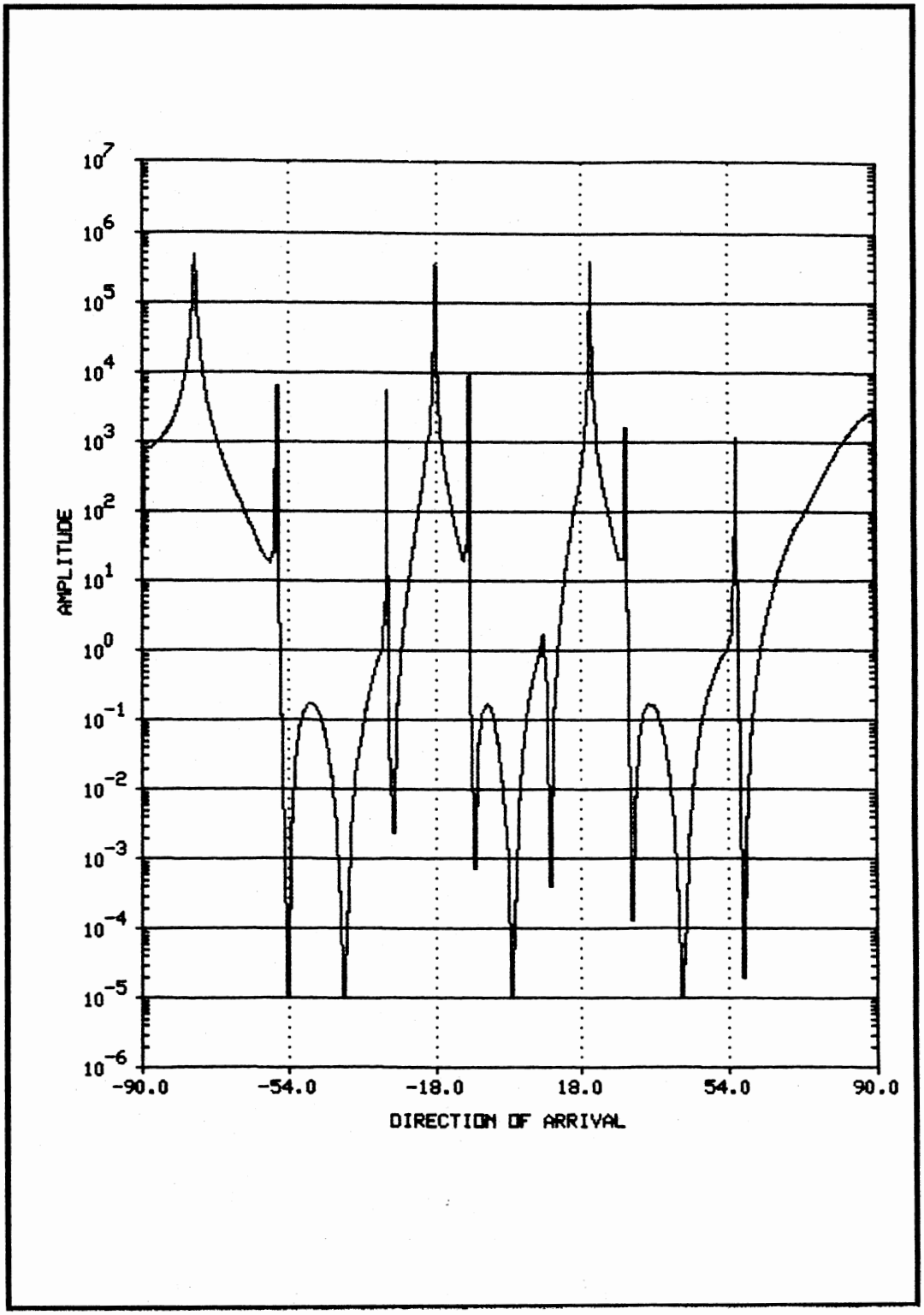


Figure 44. Simulator Driven Plot of SW Arm

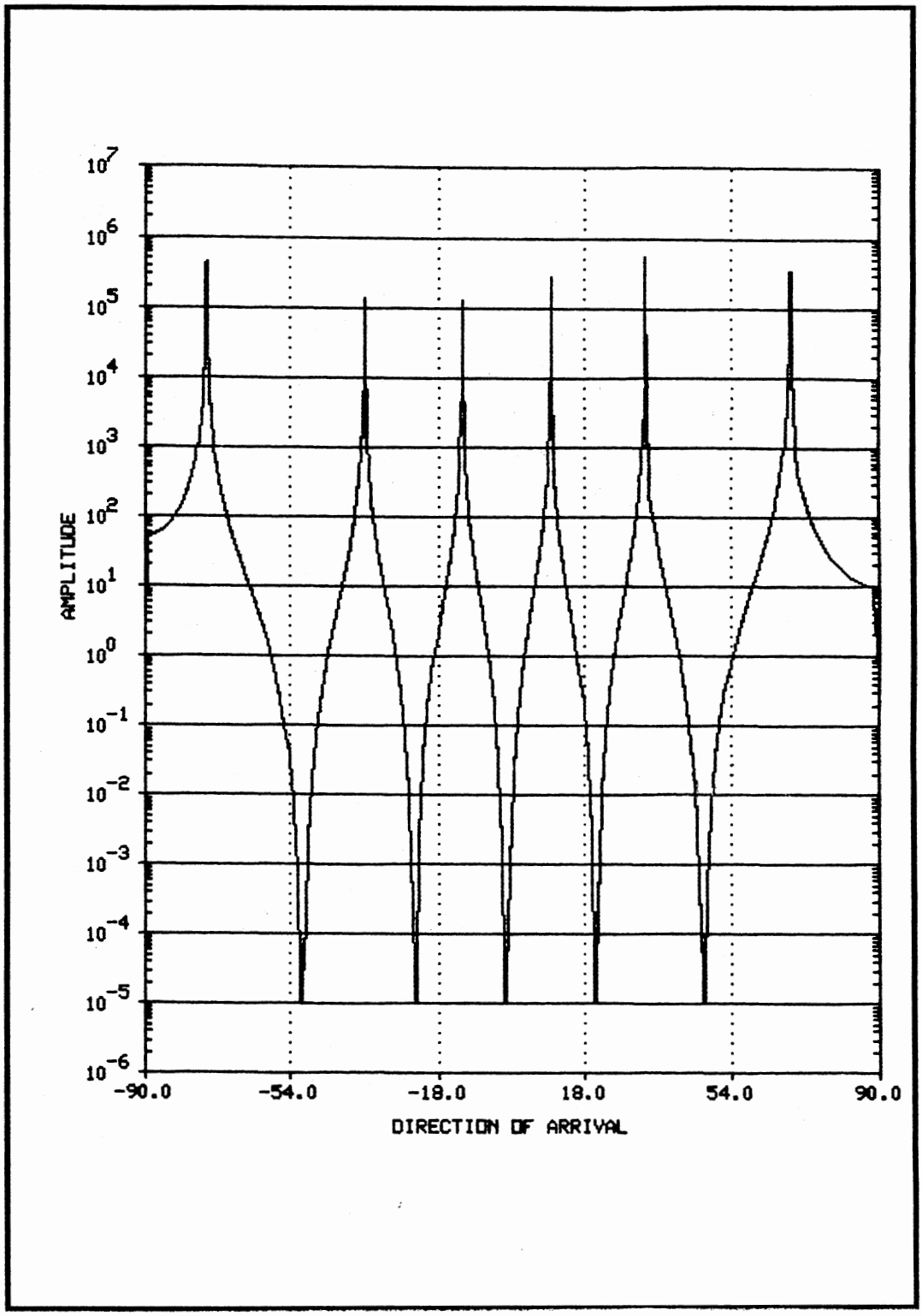


Figure 45. Simulator Driven Plot of NE/SW Arm, Center Two Elements

Simulation Results

The last set of experiments are simulations based on the same set of conditions as the first set of experiments, however these experiments are not comparisons to any known previous results.

This set is provided to examine the range of the estimator output as larger numbersof inputs and antennas are applied. They are driven by the same computer simulator in every case. Constants that are found between all of the following experiments are that the antenna array is always a linear set of omnidirectional sensors and only Gaussian noise sources are used.

Each of the experimental setups will be described in the same manner as was accomplished in the earlier set. Likewise, a brief discussion, a figure of the plotted output, and the specific estimated output DOA values will be given.

Because of the greatly improved speed of this procedure, larger arrays can be used in this section than is normally seen in reference DOA experiments. The actual times for the resulting runs are not included in the data, however they do not differ significantly from the values established in the earlier chapters. Again, the primary focus in this chapter is how well the two-vector estimator empirically performs in terms of resolution and accuracy.

Experiment Number 14

DOA	Two separate plots are provided, each with 9 overlays of the separate runs starting at 0 degrees, and every 10 degrees to 90, all at 27 dB and -6 dB.
Correlation	NA, single wavefronts
Number of antennas	64
Interelement spacing	.5λ
Number of samples	32 per source
Output	0.0, 10.1, 20.0, 30.0, 39.7, 50.0, 60.2, 70.2, 78.1, 90.0, Figure 46 2.8, 10.8, 20.2, 30.1, 39.6, 49.8, 60.9, 70.3, 79.6, 85.5, Figure 47

This experiment was to demonstrate the single arriving wavefront estimator output across a wide range of inputs varying the input amplitudes, and using a lower number of samples. Only the positive half of the DOA range is presented because the negative half was found to be essentially a mirrored performance scale of the positive half. Each run was done with a single arriving wavefront to measure the performance without the interaction of other arriving signals. Each run was then overlayed into a single figure representing all of the different runs.

This test indicates generally excellent overall performance without showing any obvious bias and demonstrating good accuracy. There is a weakness near zero degrees with low SNR, which is receiving directly broadside to the antenna array. The overlay procedure in this figure shows what appears to be a very noisy floor. What is important is the clear DOA peaks which can be identified in the output without ambiguity.

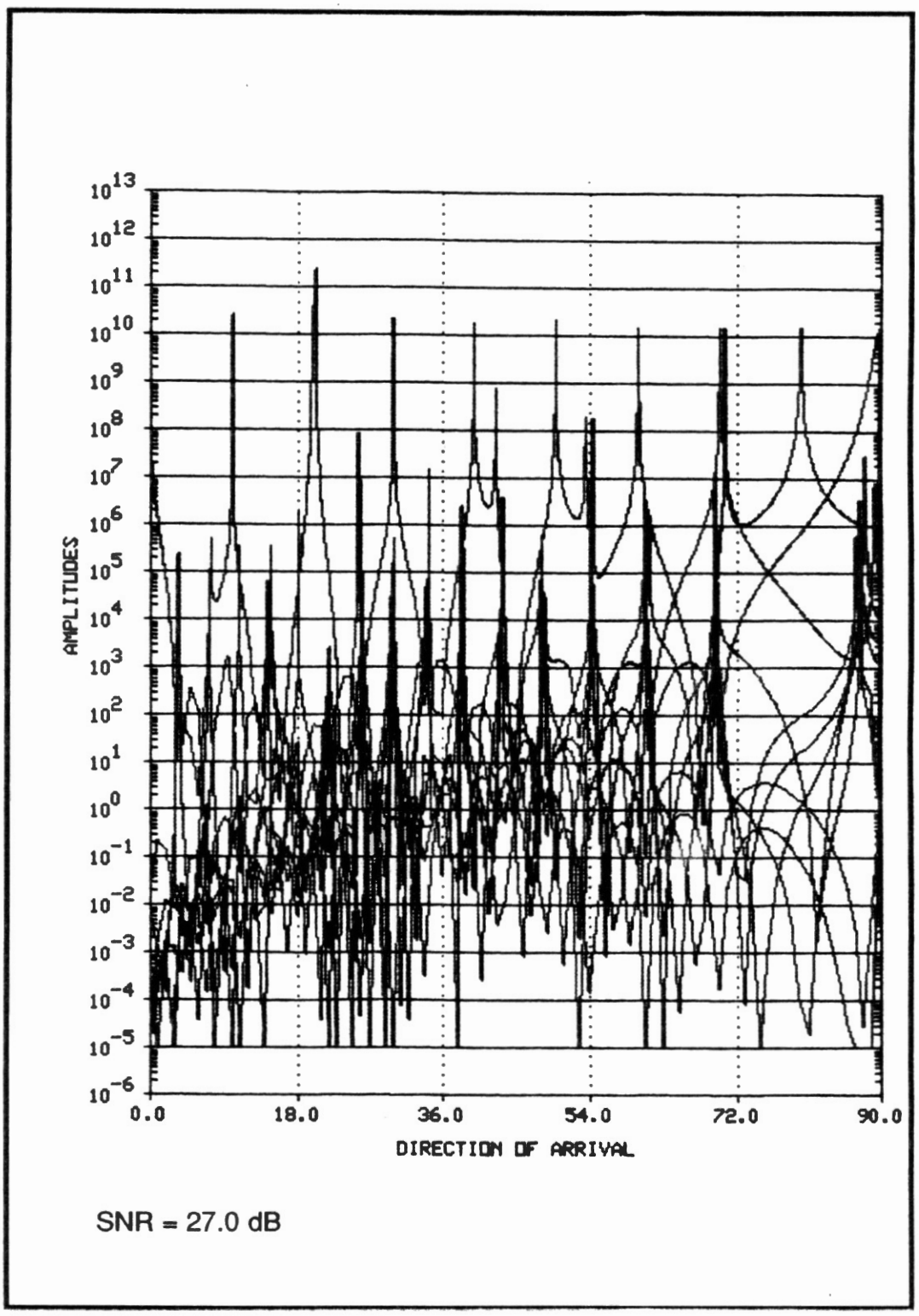


Figure 46. Experiment Number 14, Plot of $\xi(\theta)$ vs DOA Bearing

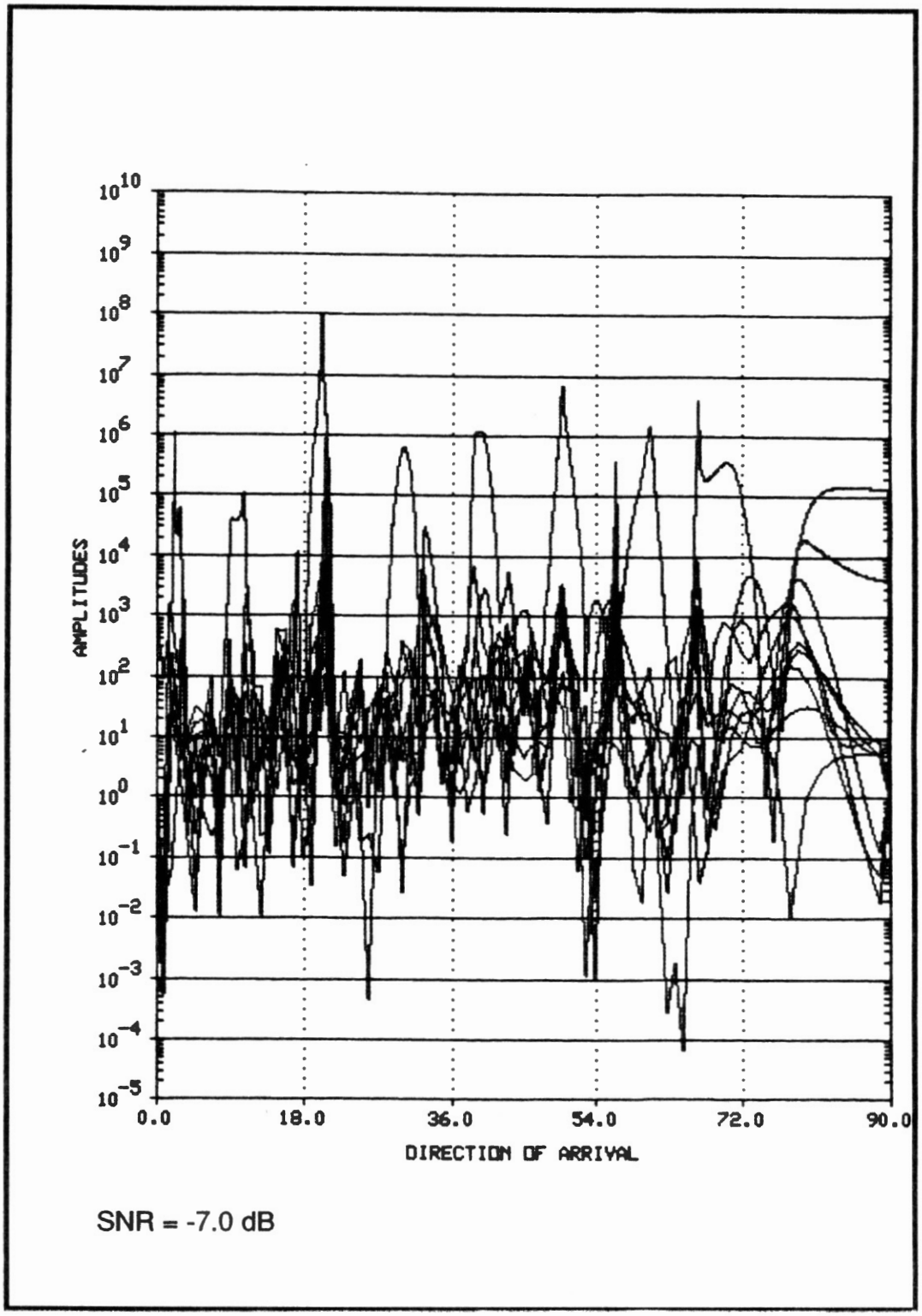


Figure 47. Experiment Number 14, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 15

DOA	22 and 24 degrees at 27, 17, 7, 0, and -7 dB
Correlation	0
Number of antennas	32
Interelement spacing	.5λ
Number of samples	32 per source
Output	22.1, 24.0, Figure 48
	22.1, 24.0, Figure 49
	22.3, 24.1, Figure 50
	22.6, 24.6, Figure 51
	23.0, Figure 52

It is seen in this data that at first as the signal strength lowers, the resolution is maintained but accuracy begins to lower. Eventually the two signals combine to a single signal at exactly the average between the two actual DOAs.

The signal peak values relatively correspond, in that the higher SNR sources have higher peak values. Although this is not normally true with all EV/EV estimators, there is obviously some corellation in this case.

The two-vector estimator correctly output that two waves were present except in the -7 dB case, where the output indicated only one wavefront. Of course, in the this case the two signals had coalesced into a single wave, hence the procedure to determine the number of arriving signals was accurate within the capacity of the estimator to resolve the signals.

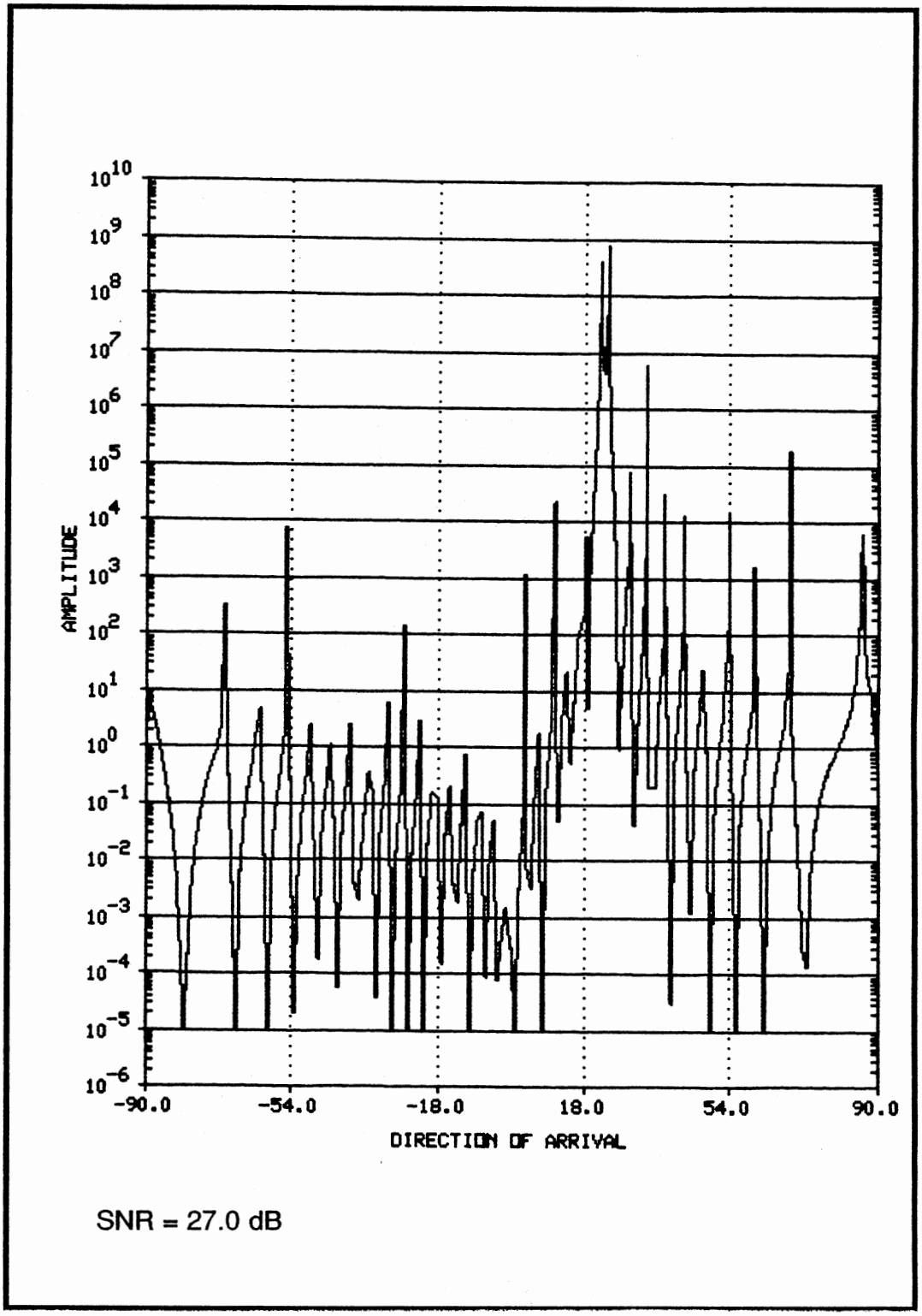


Figure 48. Experiment Number 15, Plot of $\xi(\theta)$ vs DOA Bearing

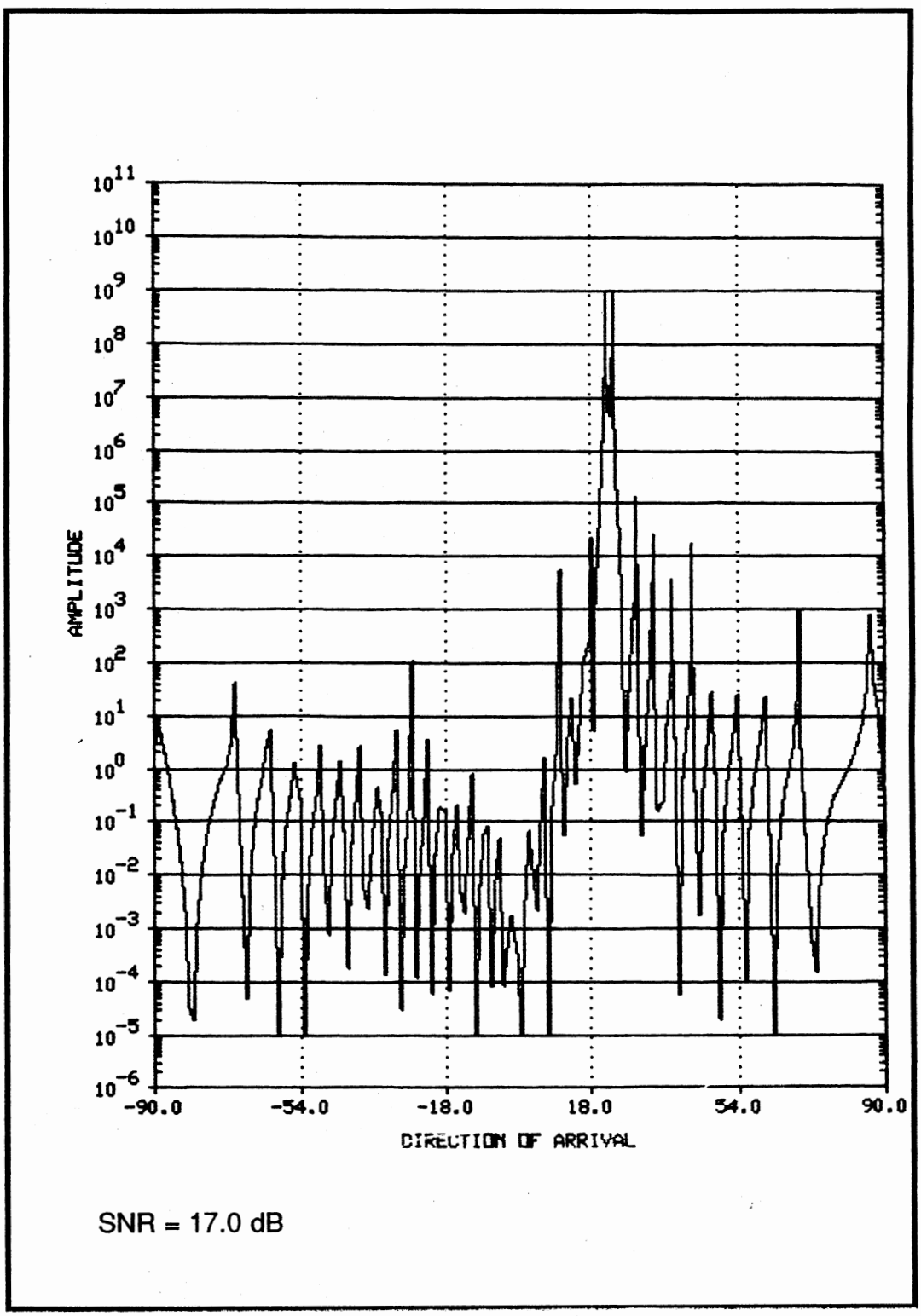


Figure 49. Experiment Number 15, Plot of $\xi(\theta)$ vs DOA Bearing

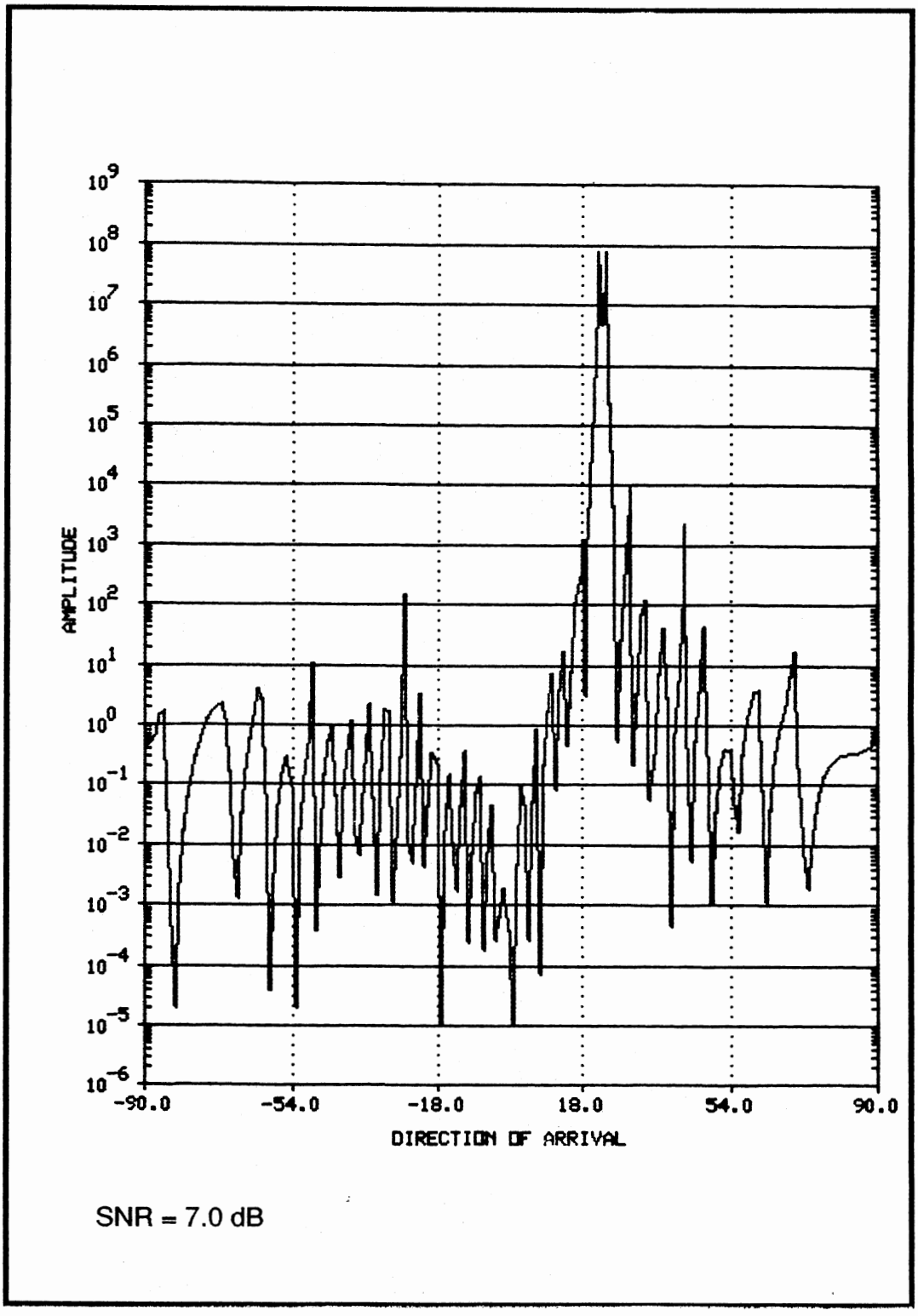


Figure 50. Experiment Number 15, Plot of $\xi(\theta)$ vs DOA Bearing

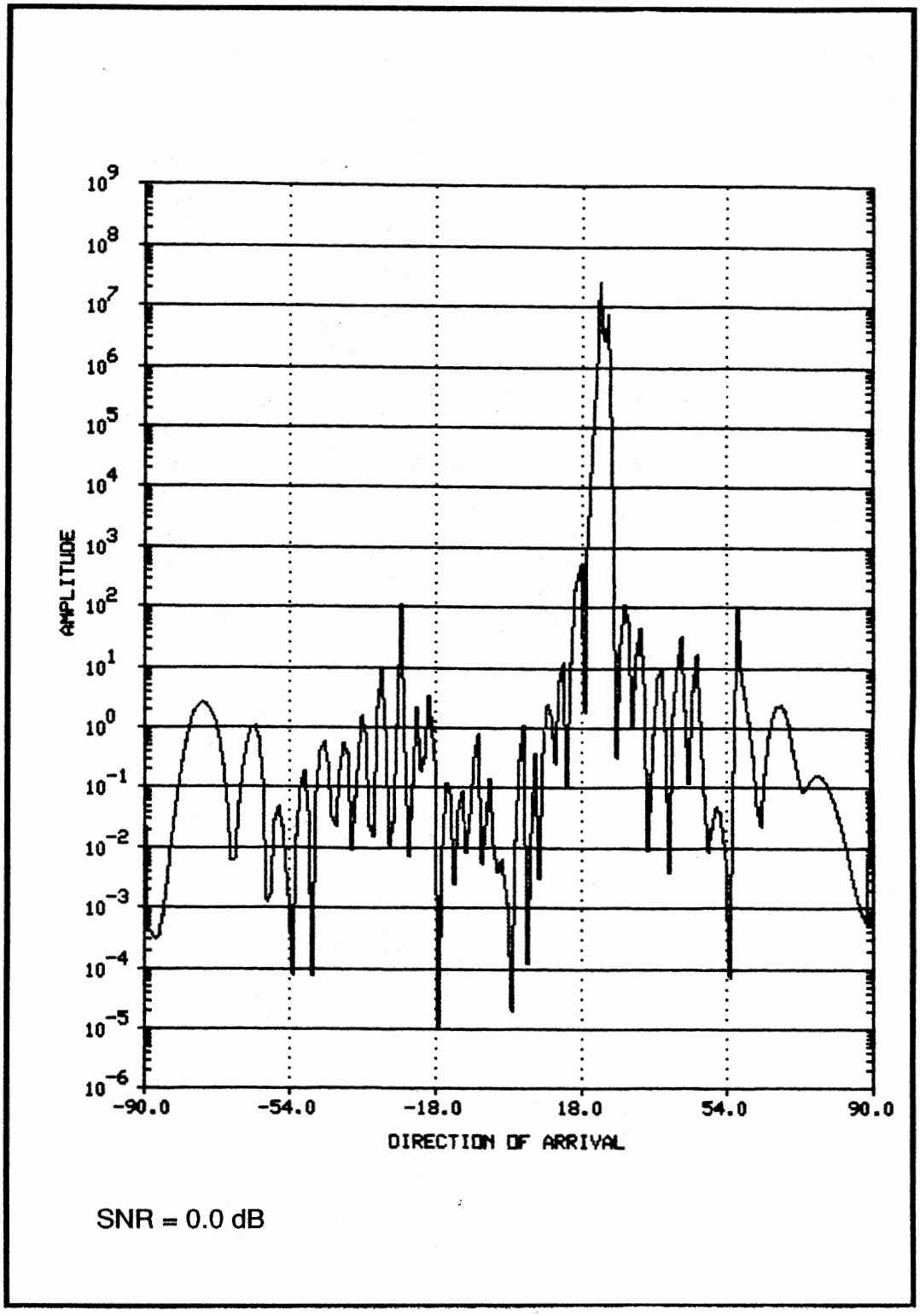


Figure 51. Experiment Number 15, Plot of $\xi(\theta)$ vs DOA Bearing

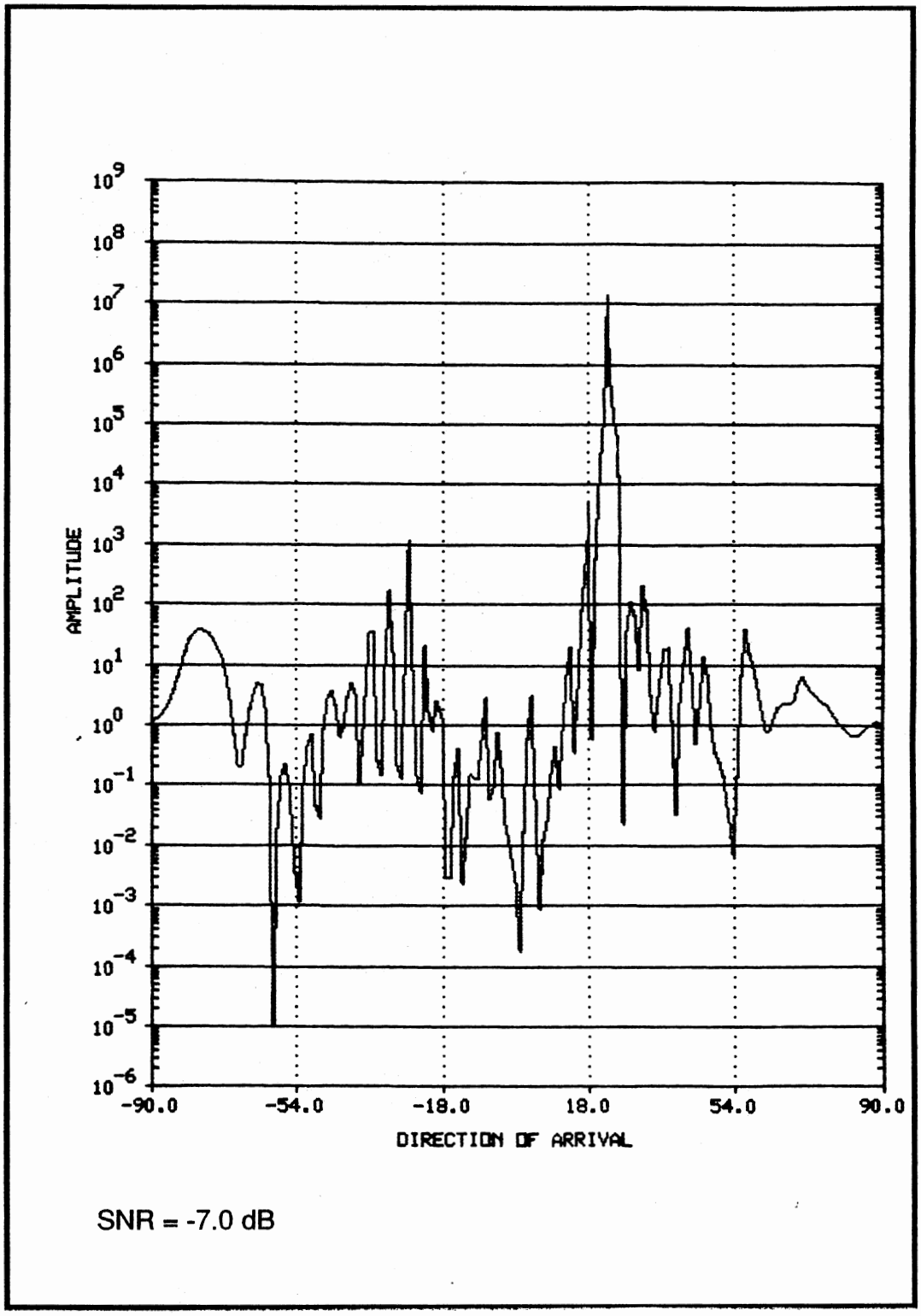


Figure 52. Experiment Number 15, Plot of $\xi(\theta)$ vs DOA Bearing

Experiment Number 16

DOA	-39.2 degrees at 10 dB, -5.7 degrees at 10 dB 0.0 degrees at 10 dB, 25.5 degrees at 10 dB, 76.6 degrees at 10 dB
Correlation	0
Number of antennas	128
Interelement spacing	.5λ
Number of samples	32 per source
Output	-39.1, -5.7, -0.6, 25.5, 76.2, Figure 53

All five signals are clearly identified, four with very high accuracy with the zero degrees showing higher but reasonable error. No bias is present.

In this case all five peaks are very distinct, and have a significantly greater magnitude than the surrounding values. The two-vector estimator correctly identified the number of arriving waves.

Experiment Number 17

DOA	-67.8, -51.7, -23.6, -5.5, 44.6, 60.5, 89.8 degrees all at 20 dB
Correlation	0
Number of antennas	16, 64, 128
Interelement spacing	0.5λ, 0.125λ, 0.0625λ
Number of samples	20 per source per antenna
Output	-67.7, -51.7, -23.5, -5.5, 44.7, 45.9, 60.4, 83.3 Figure 54 -68.2, -51.4, -23.6, -5.5, 42.5, 44.9, 60.5, 86.5 Figure 55 -68.1, -51.6, -23.6, -5.5, 42.6, 44.0, 60.5, 88.6 Figure 56

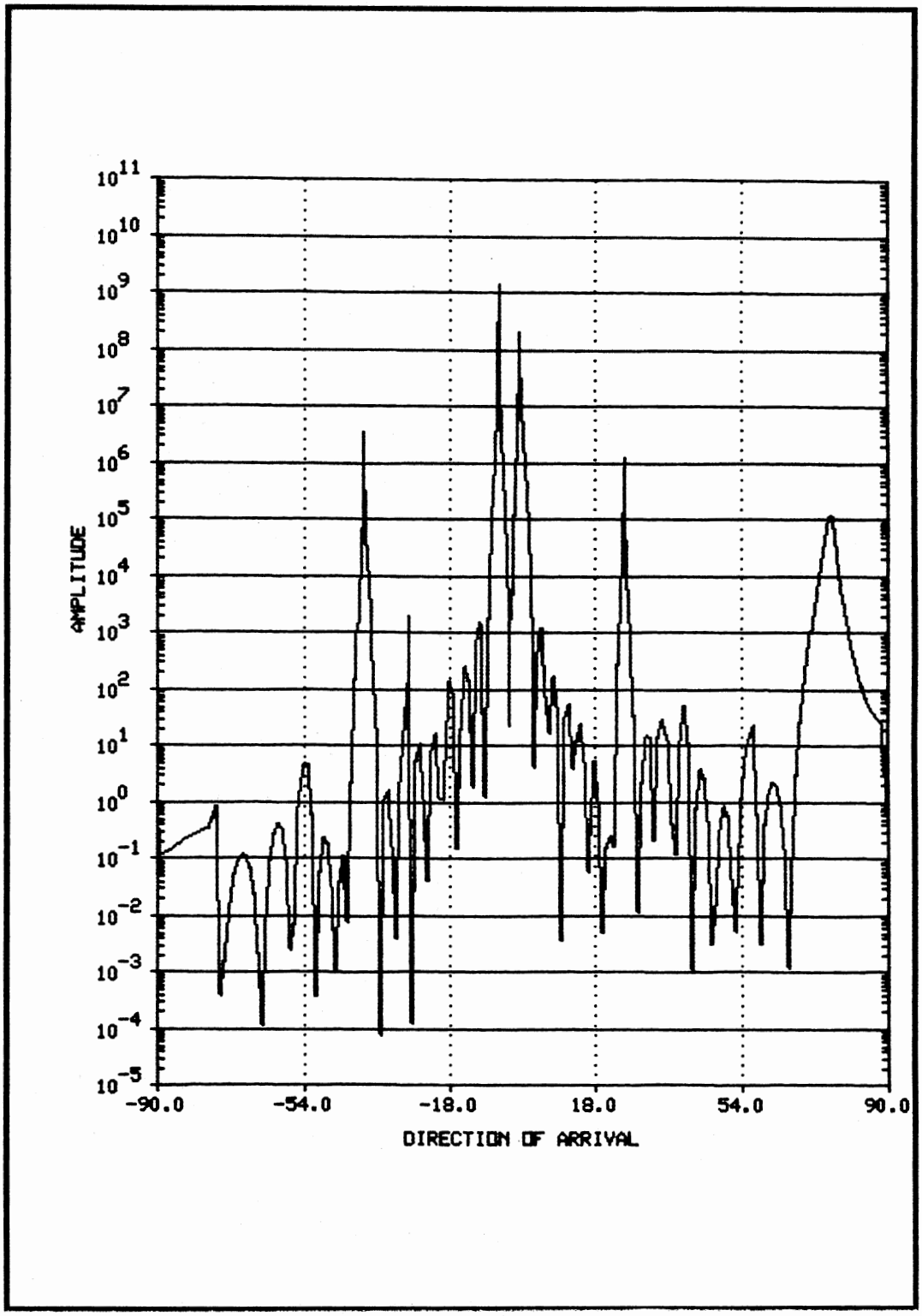


Figure 53. Experiment Number 16, Plot of $\xi(\theta)$ vs DOA Bearing

This experiment includes seven sources spread across the 180 degree range used with a linear array. The first case uses 16 antennas spaced at .5λ and 20 samples per antenna. The next case quadruples the number of antennas but keeps the aperture the same by quartering the interelement spacing. The third case doubled the antennas and halved the spacing.

As could be expected the additional samples improve the estimate accuracy and increased the hight of the peak. In particular, the peaks of the sources nearer to zero degrees were increased the most where some of the outer signal's peaks were decreased in value.

There was a splitting for the source at 44.6 degrees and two peaks were formed where only one should have appeared. This peak splitting does not occur often, but when it does it is usually has less than one degree separation, which is less than the resolution possible considering the antenna aperture. This split boarders on the resolution capability, hence they can be considered a single arriving wave, allowing the next largest peak to be added to the DOA set.

The correct value was determined for the number of arriving waves. It should be noted that when two wavefronts are as close as the split 44.6 degree reading indicates, then only a single wavefront would have been indicated by the number of arriving estimation procedure.

Many other experiments are indicated from those above. The real-time performance of the estimator allows very fast experimentation procedures to be established. It appears possible to adjust the procedures to enhance the estimate as discussed above. Some improvement would be gained, by simply tailoring the antenna array system, the procedures, and the implemented algorithms to each other.

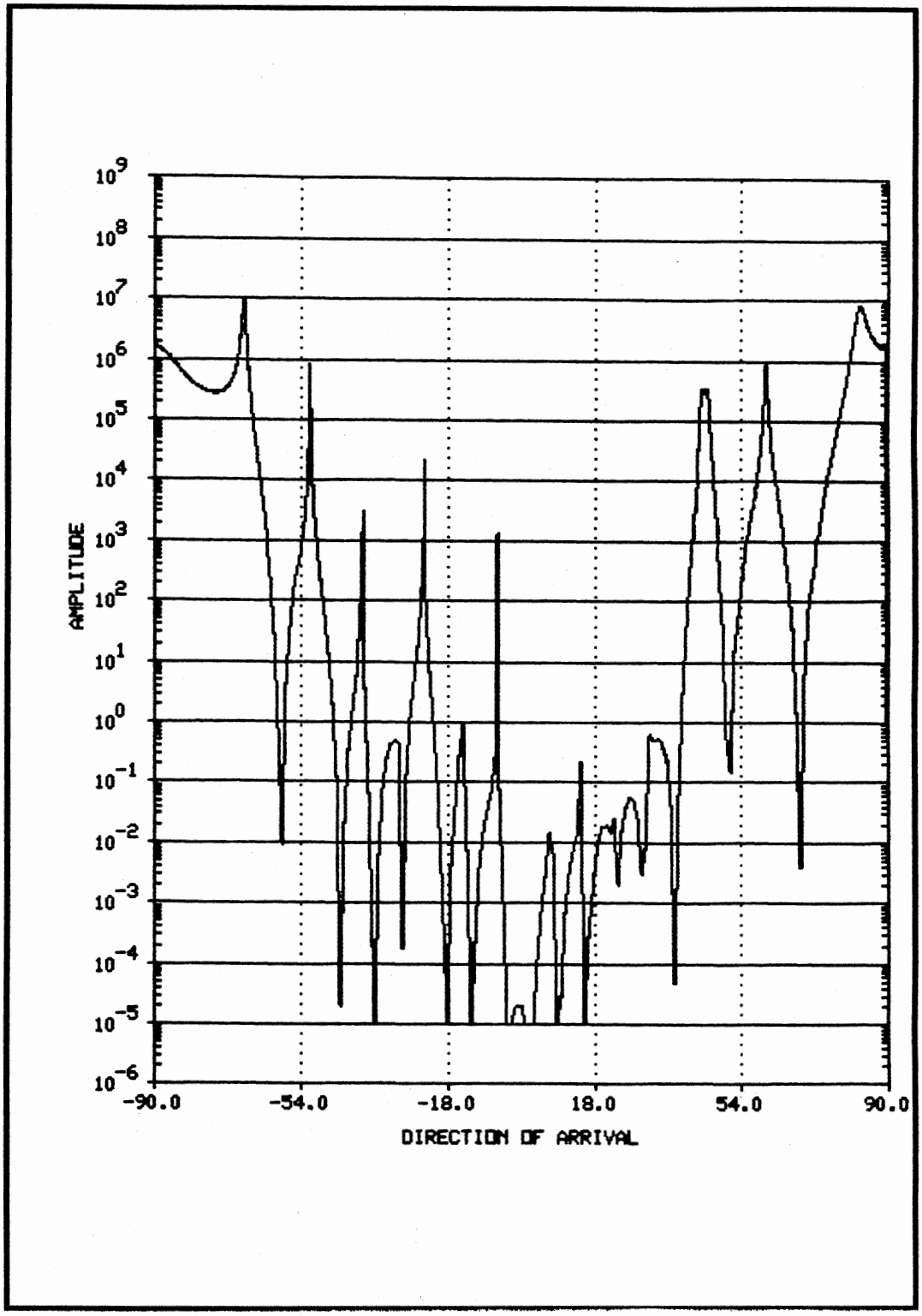


Figure 54. Experiment Number 17, Plot of $\xi(\theta)$ vs DOA Bearing, 16 Antennas, 7 Arriving Wavefronts

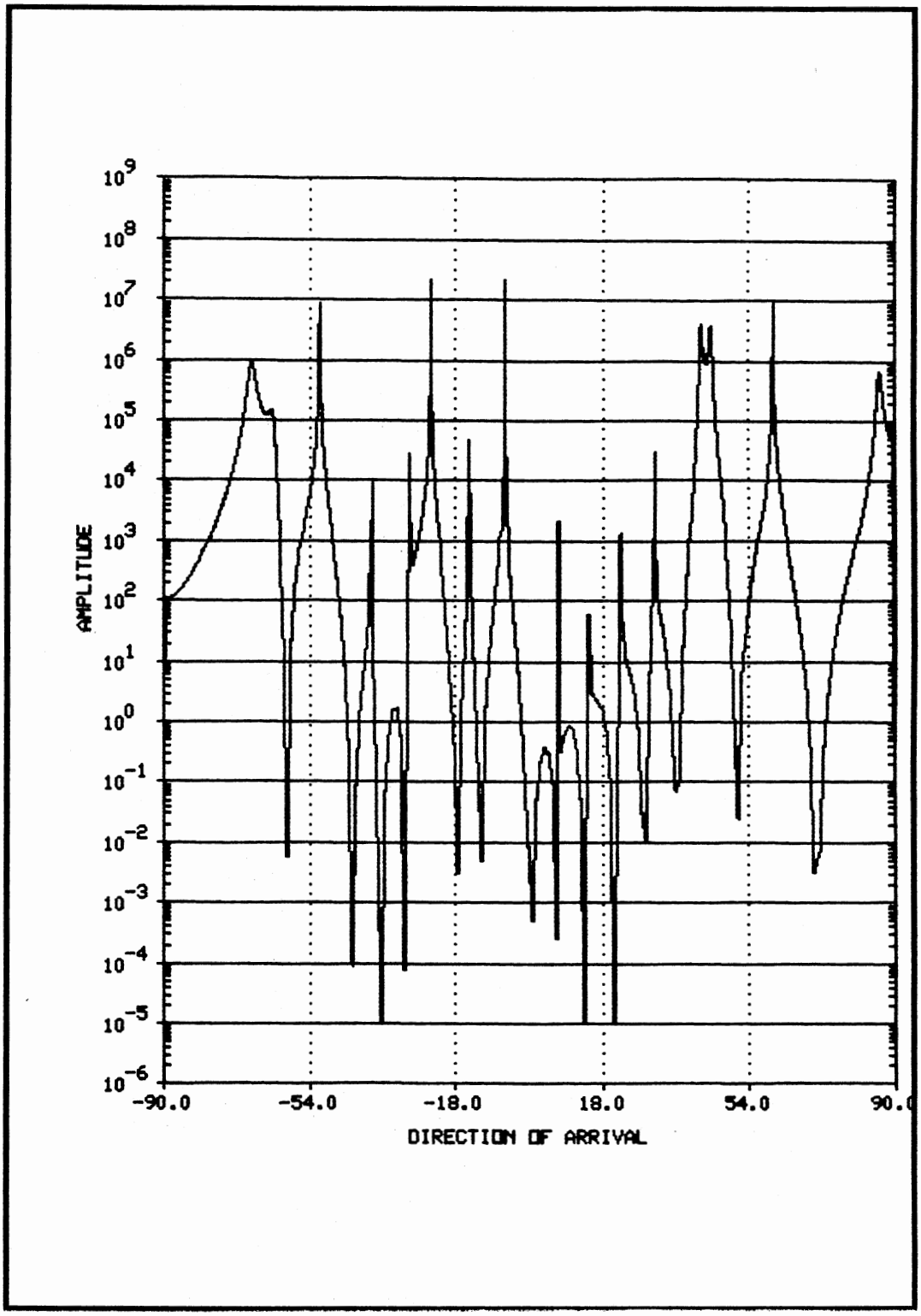


Figure 55. Experiment Number 17, Plot of $\xi(\theta)$ vs DOA Bearing, 64 Antennas, 7 Arriving Wavefronts

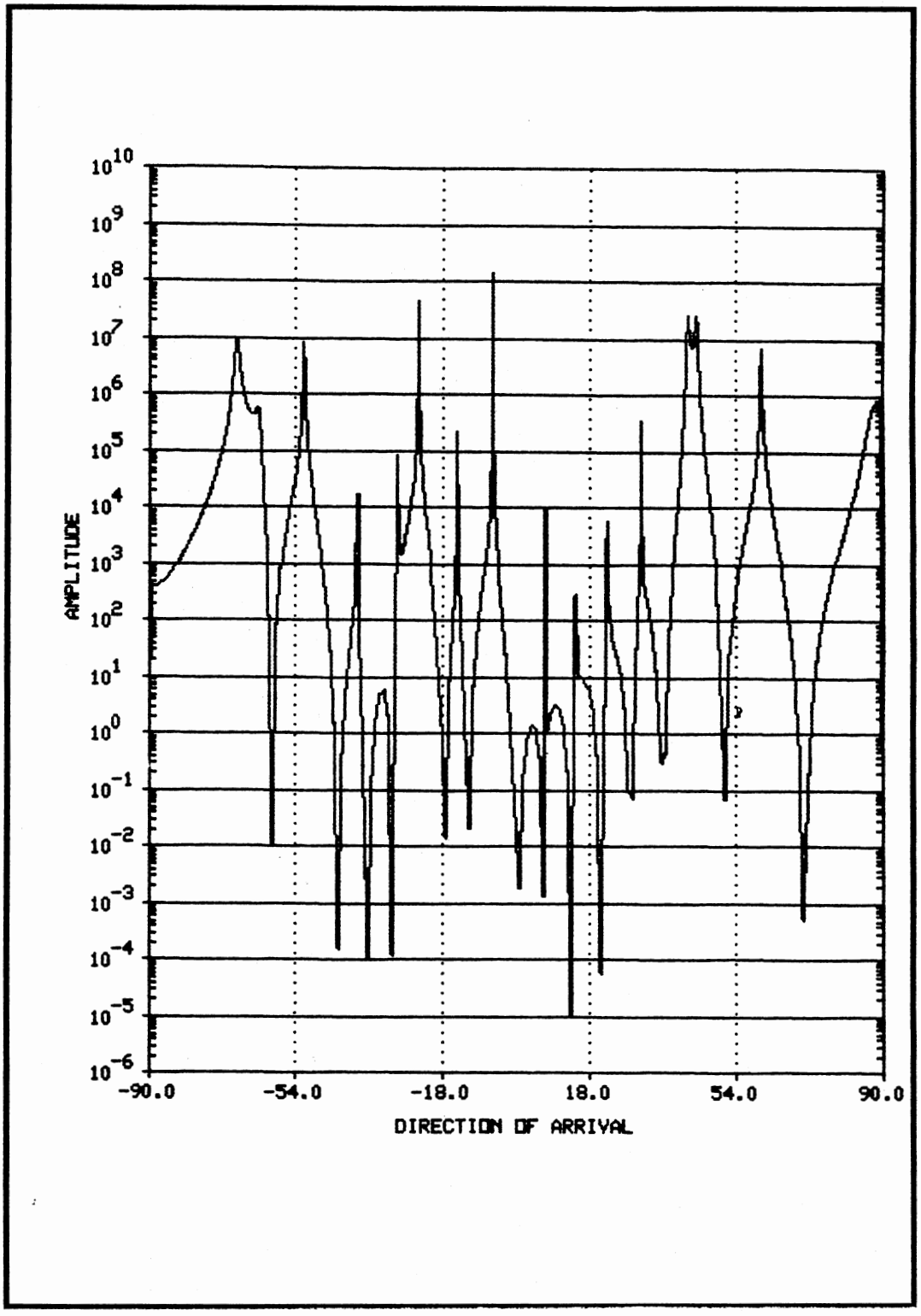


Figure 56. Experiment Number 17, Plot of $\xi(\theta)$ vs DOA Bearing, 128 Antennas, 7 Arriving Wavefronts

CHAPTER X

SUMMARY AND AREAS FOR FUTURE RESEARCH

Real-time passive bearing determination from large antenna arrays is an extremely important research area in communications, sonar, and seismic applications. The high resolution eigenvalue techniques such as MUSIC perform well in the batch mode of operation, but contribute little to on-line work.

There has been activity in modifying MUSIC and other high resolution algorithms to reduce the computational burdens and yet maintain the high resolution estimation capability. The approach in this dissertation was to apply a MIMD parallel processor against an improved two-eigenvector DOA procedure using the multi-algorithmic mode to accelerate convergence allowing dramatic decrease in the time required for computations. Thus an equivalent output is seen to be obtained, but with a significant reduction in processing time.

This work has successfully shown that the nature of the eigenstructure based algorithms are excellent candidates to use both the first level of parallelization and the advanced multi-algorithmic parallel techniques in the solution. The specific improvement in speedup allowing the new parallel algorithm to reach real-time processing speeds is a worth-while accomplishment in its own right.

The demonstration of the improved MIMD parallel processing technique of multi-algorithmic acceleration is also a contribution in that it provides a faster solution of the problem then simply trying to apply more and more processors

against multiple loops found traditional serial procedures. It also creates a unique approach for the solution to resolve the number of arriving wavefronts.

The capabilities of the developed estimator peaking function was a direct result of the efforts to minimize computations to improve speed performance and it has been shown to provide highly satisfactory outputs in most situations.

Several areas for possible future research areas can be suggested from the work completed here.

Direct improvements have already been suggested possible in speed and estimator capability by tailoring the estimator for direct application to a specific DOA system. This is possible by bypassing many of the compromises that were necessary in the prototype "one size fits all" implementation.

The method of allocation of processors for the first level parallelization and the multi-algorithmic procedure was essentially a static procedure in this study. There is room for improvement once a tailored system is developed to determine a dynamically responding workload method which would more effectively apply the parallel computer.

This work was done with a Gaussian noise model, but treated as if spatially white so that $\mathbf{R}_b$ was an identity matrix. When the noise distribution, $\mathbf{R}_b$, is known, improvements in the estimate can be obtained by developing improved methods to estimate $\mathbf{R}_b^{-1}$. This work would be especially beneficial in different environments if applied to different types of noise situations.

Advanced study relating the eigenvalue distribution, their DOAs, and the antenna array geometry would be a reasonable research direction to allow this procedure to be extended into other than colinear arrays where the number of arriving wavefronts is unknown.

The work could be expanded into a planar array or any three dimensional geometry that follows the same basic procedures.

Further advances may be possible in other areas because of the on-line performance that is available. These include the study of relationships of incoming signals in rapidly varying signal-to-noise situations, a moving source relation to range estimation, and variation in the types of noise during sampling.

Another possibility of future research is associated with the cancellation of the incoming signals using a modified version of the developed two-vector estimator. Experiments have shown the possible existence of the capability to separate the signal from noise in a highly colored (directional) noise environment. Although not pursued here, there were some indications that it would be possible to cancel or mask a signal or directional noise when desired with a recusive adjustment to the input data.

In sonar applications varying propagation speed of the signal as an input parameter along with temperature and medium density variations might allow additional information to be obtained about the environment when known control signals are present.

There were indications that this DOA system can respond to jammer interference, or that it could provide a base for building a parallel adaptive recursive system that processes and updates by sample interval.

There seems to be no reason that this method could not be equally improved by applying some of the preprocessing decorrelation procedures to the sample covariance matrix to improve signal detection for coherent signals. Likewise, advances in broadband analysis which is also a computationally burdened preprocessing procedure may find extension of this work valuable and natural.

Given an on-line system, there is a significant potential in the character of these areas to yield several improvements in the above research areas. The problems are complex, but considering the many areas of application, the solutions will have very broad physical application with their finalization.

REFERENCES

Alsup, J.M. (1984). High-resolution techniques for two-dimensional estimation of angle-of-arrival for planar array. Proceedings IEEE ,Acoustics, Speech, and Signal Processing 84, (84), 40.1.1-40.1.4

Anton, H. (1977). Elementary Linear Algebra. New York: John Wiley & Sons.

Asbury, R., Frison, Steven G., and Roth, T. (September 1985) Concurrent computers ideal for inherently parallel problems. Santa Clara, CA: Intel Corporation.

Blahut, Richard E. (1985). Fast Algorithms for Digital Signal Processing. Reading, Ma: Addison-Wesley Pub. Co. Processing II, 826, 12-16.

Bond, J. (1987). Parallel-processing concepts finally come together in real systems. Computer Design , 51-74.

Bresler, Yoram, and Macovski, A. (1986). On the number of signals resolvable by a uniform linear array. IEEE Transactions on Acoustics, Speech, and Signal Processing, 34 (6), 1361.

Brogan, W.L. (1985). Modern Control Theory, Second Edition, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Bromley, Keith, Speiser, Jeffery M., and Whitehouse, Harper J. (1985). Advanced Real-Time Parallel Signal Processing Architectures, Algorithm Implementations, Applications. San Diego, Ca: Evolving Technology Inst.

Bronez, Thomas P., and Cadzow, James A. (1983). An algebraic approach to superresolution array processing. IEEE Transactions on Aerospace and Electronic Systems, 19, (1), 123-133

Buckley, Kevin M., and Griffiths, LLoyd J. (1988). Broad-Band Signal-Subspace Spatial-Spectrum (BASS-ALE) Estimation. IEEE Transactions on Acoustics, Speech, and Signal Processing, 36,(7), 953-964.

Cadzow, James A. (1988). A high resolution direction-of-arrival algorithm for narrow-band coherent and incoherent sources. IEEE Transactions on Acoustics, Speech, An Signal Processing, 36 (7), 965-979.

Cadzow, James A., Kim, Young-Soo, and Shiue, Dong-Chang (1987). Signal eigenvector method for general array signal processing. SPIE Advanced Algorithms and Architecture for Signal Processing II, 826, 56-72.

Conte, S.D. and de Boor, C. (1980). Elementary Numerical Analysis An Algorithmic Approach, Third Edition. New York: McGraw-Hill Book Co.

Denning, P. (1986). The science of computing parallel computation. American Scientist, 73.

Eberlein, P.J. (1987). The parallel solution of eigenproblems on multiprocessor systems. SPIE Advanced Algorithms and Architectures for Signal Processing II 826, 146-151.

Egan, M. (1988). Parallel processing capabilities added to IBM 3090s. MIS Week, 9, (9).

Flynn, M.J. (1972). Some computer organizations and their effectiveness. IEEE Trans. Comp., C-21, ,948-960.

Flynn, M.J. (1966). Very high-speed computing systems. IEEE Proceedings, 54, 1901-09.

Fox, G. C. and Otto S. W. (1987). Matrix algorithms on a hypercube I: Matrix multiplication. Parallel Computing 4,17-31.

Fuhrman, Daniel R. (1987). Adaptive MUSIC. SPIE Advanced Algorithms and Architectures for Signal Processing II 826, 92-95.

Gourlay, A. R., and Watson, G. A. (1973). Computational Methods for Matrix Eigenproblems. New York: John Wiley & Sons.

Grunwald, Dirk C., and Reed, Daniel A. (1986). Benchmarking hypercube hardware and software. Report No. UTUCDCS-R-1303, Urbana, IL: Dep of Computer Science, University of Illinois.

Haykin, S. (Editor), Justice, J.H., Owlsey N.L., Yen J.L., and Kak, A.C. (1985). Array Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Huang, K. and Briggs F. (1980). Computer Architecture and Parallel Processing. New York: McGraw-Hill Book Co.

Intel Scientific Computers (1987). iPSC/2 Green Hlls FORTRAN language reference manual. Beaverton, OR.

Intel Scientific Computers (1987). iPSC/2 User's guide. Beaverton, OR.

Intel Scientific Computers (1986). IPSC Concurrent Programming Workshop. Beaverton, OR.

Ipsen, Ilse C.F., Sasd Youcef (1985). The Impact of parallel architectures of the solution of eigenvalue problems. Large Scale Eigenvalue Problems, Proceedings of the IBM Europe Institute Workshop on Large Scale Eigenvalue Problems Held in Oberlech, Austria, July 8-12, 1985, The Netherlands: Elsevier Science Publishers B.V., 37-49.

Johnson, Don H. (1982). The application of spectral estimation methods to bearing estimation problems. Proceedings Of The IEEE 70, (9), 1018-28.

Johnson, Don H., and DeGraaf, Stuart R. (1982). Improving the resolution of bearing in passive sonar arrays by eigenvalue analysis. IEEE Transactions On Acoustics, Speech, And Signal Processing ASSP-30(4), 638-47.

Kaplan, David J. (1987). High resolution direction finding algorithm development. Progress report, System Planning Corporation

Karp, A.H. (1987). Programming for parallelism. Computer, 43-57.

Kriel, Charles W. (1988). Lp-norm estimation techniques applied to multiple emitter location. Ph.D. Dissertation, Oklahoma State University.

Kumaresan, Ramdas, and Shaw, Arnab K. (1988). Superresolution by Structured Matrix Approximation. IEEE Transactions on Antennas and Propagation, 36 (1), 34-44.

Kumaresan, Ramdas, and Tufts, Donald W. (1983). Estimating the angles of arrival of multiple plane waves. IEEE Transactions on Aerospace and Electronic Systems, 19 (1), 134-8.

Luk, Franklin T., and Park, H. (1987). On the equivalence and convergence of parallel Jacobi SVD algorithms. SPIE Advanced Algorithms and Architectures for Signal Processing II 826,152-159.

McBryan, O.A., and Van De Velde, E.F. (1987). Hypercube algorithms and implementations. SIAM J. Sci. Statist. Comput., 8.(2), 227-287.

Marple, S. Lawrence,Jr. (1987). Digital Spectral Analysis with Applications. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Marple, S. Lawrence, Jr. and Friedlander, B. (1987). Sonar signal processing on the SAXPY MATRIX 1 parallel processor. SPIE Advanced Algorithms and Architectures for Signal Processing II 826, 73-81.

Martin, G. E. (1987). Signal subspace processing of actual radio and acoustic data. Martin Analysis Software Technology, Inc. San Diego, CA.

Monzingo, Robert A. (1980). Introduction to Adaptive Arrays. New York: John Wiley & Sons.

Morris, D.J. (1977). Introduction to Communication Command and Control Systems. Oxford, England: Pergamon Press/

Owsley, Norman L. (1981). Time delay estimation in a sensor array. Transactions on Acoustics, Speech, And Signal Processing 29 (3), 519.

Parlett, B. N. (1980). The Symmetric Eigenvalue Problem. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Paulraj, Arogyaswami, and Kailath, T. (1986). Eigenstructure methods for direction of arrival estimation in the presence of unknown noise fields. IEEE Tran. on Acoustics, Speech, And Signal Processing, 34 (1), 13-20.

Pisarenko, V. F. (1973). The retrieval of harmonics from a covariance function. Geophys. J. Roy. Astron. Soc., 33. 247-266

Rattner, J. (1985). Concurrent processing: A new direction in scientific computing. AFIPS Conference Proceedings 54.

Reddi, S. S. (1979). Multiple source location - a digital approach. IEEE Transactions On Aerospace and Electronic Systems ,15(1), 95-105

Reddy, V. U. (1982). Least squares type algorithm for adaptive implementation of Pisarenko's harmonic retrieval method. IEEE Transactions on Acoustics, Speech, And Signal Processing, 30 (3), 399-405.

Reilly, James P. (1987). A real-time high-resolution technique for angle-of-arrival estimation. Proceedings of the IEEE, 75 (12),1692-4.

Schendel, U. (1984). Introduction to Numerical Methods for Parallel Computers. Chichester England: Ellis Horwood Limited.

Schmidt, Ralph O. (1981). A Single Subspace Approach to Multiple Emitter Location and Spectral Estimation. Ph.D. Dissertation, Stanford University.

Seymore, L.P.H.K., Cowan, C.F.N., and Grant P.M. (1987). Bearing estimation in the presence of sensor positioning errors. 1987 International Conference On Acoustics, Speech, And Signal Processing, 4, 2264-67.

Shaw, Arnab K., and Kumaresan, R. (1987). Estimation of angles of arrivals of broadband signals. International Conference On Acoustics, Speech, And Signal Processing 4, 2296-99.

Speiser, Jeffery M. (1985). Progress in eigenvector beamforming. Real Time Signal Processing VIII , 564.

Speiser, Jeffery M. (1987). Some observations concerning the ESPRIT direction finding method. SPIE Advanced Algorithms and Architectures for Signal Processing II, 826, 178-185.

Tufts, Donald W., and Melissinos, Constatinos D. (1986). Simple, effective computation of principal eigenvectors and their eigenvalues and application to high-resolution estimation of frequencies. IEEE Transactions on Acoustics, Speech, And Signal Processing ASSP-34 (5), 1046-53.

Wax, M., and Kailath, T. (1985). Detection of signals by information theoretic criteria, IEEE Transactions on Acoustics, Speech, And Signal Processing ASSP-33 387-392.

Wax, M., Shan, T.J., and Kailath, T. (1982). Location and the spectral density estimation of multiple sources. Proceedings of 16th Asilomar Conf. Cir., System, Comp.

Wilkinson, M. A. (1965). The Algebraic Eigenvalue Problem. London: Oxford University Press.

Williams, R.T., Prasad, S., Mahalanabis,A.K., and Sibul, L.H. (1988). An improved spatial smoothing technique for bearing estimation in a multipath environment. IEEE Transactions on Acoustics, Speech, And Signal Processing ASSP-36 (4), 425-431.

APPENDIXES

APPENDIX A

HOST FORTRAN COMPUTER PROGRAM

This appendix provides the FORTRAN-386 Computer program that is the host program for the parallel processor. The main functions of the host program are to accept the user inputs to create the simulated situations, accept an input that was the result of another simulated or real situation, prepare the data for node processing, and output the results from the nodes. Except for some longer lables used, the coded portion is in FORTRAN-77. The FORTRAN-386 implements the ANSI FORTRAN-77 (Full Language) Standard, ANSI X3.9-1978. It also implements all of the extensions to FORTRAN-77 documented in the Berkeley 4.2BSD f77 documentation and many of the undocumented extensions in the 4.2BSD f77 implementation. The iPSC/2 Green Hills FORTRAN Language Reference Manual should be consulted if computer language questions arise while reading the implemented code.

The actual computations of the DOA estimate is accomplished in the node program in Appendix B and the host program is considered an off-line process that feeds the parallel program. The host program loads the node programs and any input data necessary, and provides all of the user input and output interaction.

The following description of the code is to provide a simple guide through the process. The reader must be very familiar with the problem being solved and the language used.

A MAIN program is used which is a collection of subprogram CALLs. Each subprogram will be briefly described in the order that they appear in MAIN.

SETPID is a unique parallel processor function that sets the process ID of the program in the host so that communication from the nodes can be received.

INITIAL is a initializing subprogram that specifies the cube size to be used, establishes values for the random number generator seeds, and determines if data is to be simulated here or input from a simulated or real situation.

INSIG requires the user to provide the number of sources, and the simulated signal source parameters. The parameters include DOA, power level, initial phase and wavelength of each signals to be simulated. For a real data situation only the wavelength is required.

INANT accepts a description of the antenna array. This possible inputs are the separation between elements, and the number of elements.

INNOISE requests the user to identify the noise generators and power level for each antenna whose outputs will be combined with the simulated source signal. Possible noise generators available are Gaussian, uniform, Rayleigh, Laplacian, and impulse distributions.

GENOISMTX creates the simulated noise samples defined by INNOISE.

SAMP combines the information from INSIG, INANT, and the output from GENOISMTX to create the simulated antenna samples for the experiment.

ESNOISE builds another noise matrix which uses the same distribution as the simulated choice in INNOISE above, but uses different seeds. This is used when the generalized eigenvalue problem is solved.

NOEXPECT completes the estimation process of the noise to build the matrix $\mathbf{R}_b$ as described in the MUSIC algorithm.

INVERT creates $\mathbf{R}_b^{-1}$ when this procedure is necessary.

CREATEFILE builds an output file that contains all of the input information.

EIGPOWER interacts with the nodes to solve the power method problem.

OUT sends the results to the user.

At this point another experiment can be run again using the same parameters, using new samples from the same situation altering the number of samples, or stop.

```
      PROGRAM HOST
C
C*** BEGIN SIMULATION GENERATION PROGRAM.  ELSE READ REAL DATA
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L
C
      INTEGER ISEED1,ISEED2,SIM
C
      REAL    SIZE,ANMSIG,NMAT,NMSMP,WVLTH,
     >        ASIGPAR(159,6),
     >        SEP,ATVEC(160,5),ANOMTX(320,320),
     >        ARSGVEC(320,2),
     >        SMPVEC(320,320),S,
     >        ESMPMTX(320,320),VEC(320,2),MTX(320,320),
     >        EINOMTX(320,320),ENOMTX(320,320),
     >        AENOMTX(320,320),AIENOMTX(320,320),
     >        EIG,EIGVEC(320,2),
     >        OUTMSG(1834)
C
C   MAIN HOST PROGRAM FOR TWO-EIGENVECTOR METHOD DOA
C
      CALL SETPID(100)
C
      CALL INITIAL(SIZE,ISEED1,ISEED2,SIM)
C
      CALL INSIG(WVLTH,ANMSIG,ASIGPAR,SIM)
C
      CALL INANT(SEP,ATVEC,NMAT,SIM)
C
      CALL INNOISE(NMAT,ATVEC,SIM)
C
200   CALL GENOISMTX(NMAT,NMSMP,ATVEC,ANOMTX,ISEED1,SIM)
C
      CALL SAMP(NMAT,NMSMP,ANOMTX,SMPVEC,SEP,ANMSIG,ASIGPAR,SIM)
C
      CALL ESNOISE(NMAT,NMSMP,ATVEC,ANOMTX,ISEED2,SIM)
C
      CALL NOEXPECT(NMAT,NMSMP,ANOMTX,AENOMTX,SIM)
C
      CALL INVERT(NMAT,AENOMTX,AIENOMTX,SIM)
C
C       CALL CREATEFILE(NMAT,NMSMP,ATVEC,ANOMTX,AENOMTX,AIENOMTX,
C     >                        SMPVEC,SEP,ARSGVEC,SIM)
C
C AT THIS POINT ALL INPUT DATA AND ALL SIMULATED SAMPLES
C OF THE SIGNALS AND NOISE HAVE BEEN COMPUTED. FROM THIS
C POINT, THE TIMING OF THE SOLUTION CAN BE COMPARED WITH
C THE PERFORMANCE OF OTHER TECNIQUES.  IF ACTUAL DATA IS
C AVAIALBLE THIS IS THE POINT IT WOULD BE MADE AVAILABLE
C TO MUSICPOWER AS 'SMPVEC'.
C
100   CALL EIGPOWER(SIZE,WVLTH,SEP,NMSMP,SMPVEC,AIENOMTX,
     >                AENOMTX,NMAT,OUTMSG,SIM)
C
      CALL OUT(OUTMSG)
C
```

```
      PRINT*, 'REPEAT WITH SAME(0), DIFF(1), STOP(2)'
      READ*,I
      IF(I.EQ.0)GO TO 100
      IF (I.EQ.2)GOTO 2020
      PRINT *,'CHANGE SEED, NO(0), YES(#) ?'
      READ *,K
      IF(K.NE.0)ISEED1=K
      SIM=1
      GO TO 200
2020  CLOSE(2)
      CLOSE(1)
      CLOSE(15)
      END
C
C**********************SUBROUTINE INITIAL*******************************
C
      SUBROUTINE INITIAL(SIZE,ISEED1,ISEED2,SIM)
C
      IMPLICIT NONE
      INTEGER I,J,K,L
C
      INTEGER ISEED1,ISEED2,SIM
C
      REAL SIZE1,SIZE
C
      PRINT *,'SIMULATION BY THIS PROGRAM(1) OR'
      PRINT *,'DATA FILE TO BE READ'
      READ *,SIM
C
      IF (SIM.EQ.0) THEN
C
          OPEN(3,NAME='input.real',STATUS='OLD',ERR=999)
C
          OPEN(4,NAME='input.imag',STATUS='OLD',ERR=999)
C
      ENDIF
C

      OPEN(1,NAME='in.dat',STATUS='NEW')
C
      OPEN(2,NAME='out.dat',STATUS='NEW')
C
      OPEN(15,NAME='run.dat',STATUS='NEW')
C
      SIZE1=1
      ISEED1=12357
      PRINT *,'CHANGE SEED, NO(0), YES(#) ?'
      READ *,I
      IF(I.NE.0)ISEED1=I
      ISEED2=13357
      PRINT *,'WHAT CUBE SIZE (1-',SIZE1,') DO YOU WISH'
      READ  *,SIZE
      WRITE (1,100)SIZE
100   FORMAT(' SIZE OF CUBE BEING USED =',F3.0/)
C
1000  RETURN
999   PRINT *,'INPUT FILE NOT AVAILABLE, USING SIMULATION'
      SIM=1
```

```
      GOTO 1000
C
      END
C
C**********************SUBROUTINE INSIG*********************************
C
      SUBROUTINE INSIG(WVLTH,ANMSIG,ASIGPAR,SIM)
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L,SIM
C
      REAL ANMSIG,ASIGPAR(159,6),TDOA,TST,TPH,WVLTH
C
      IF (SIM.EQ.0)THEN
          PRINT *, 'THIS IS NOT A SIMULATION, REAL DATA BEING USED.'
          PRINT *
          PRINT *, 'YOU MUST PROVIDE THE WAVELENGTH IN METERS.'
          READ  *,WVLTH
          WRITE(2,5000)I,WVLTH
          WRITE(1,5000)I,WVLTH
          ASIGPAR(I,4)=WVLTH
          GOTO 6000
      ENDIF
      PRINT*
      PRINT *,'INPUT THE SIGNAL(S) CHARACTERISTICS AT THIS POINT.'
      PRINT *
      PRINT *,'HOW MANY SOURCES WILL THERE BE  ?'
      READ  *,ANMSIG
      WRITE(2,1000)ANMSIG
      WRITE(1,1000)ANMSIG
1000  FORMAT(' ACTUAL NUMER OF SIGNALS= ',F3.0/)
      PRINT *,'FOR EACH SIGNAL INPUT THE'
      DO 10 I=1,ANMSIG
          PRINT *,' DIRECTION OF ARRIVAL FOR SIGNAL ',I,
     >            ' IN DEGREES ?(-90 TO +90 OFF NORMAL TO ARRAY)'
          READ  *,TDOA
          WRITE(2,2000)I,TDOA
          WRITE(1,2000)I,TDOA
2000      FORMAT('DIRECTION OF ARRIVAL FOR SIGNAL',I3,' IS ',F6.2,
     >           ' DEGREES'/)
          TDOA=TDOA*.01745329252
          ASIGPAR(I,1)=TDOA
          PRINT *,' SIGNAL ',I,'IN DB RELATIVE TO NOISE POWER'
          READ  *,TST
          TST=(2.0*10.0**(TST/10.0))**.5
          WRITE(2,3000)I,TST
          WRITE(1,3000)I,TST
3000      FORMAT('SIGNAL STRENGTH FOR SIGNAL',I3,' IS ',F6.2,
     >           ' MICROWATTS'/)
          ASIGPAR(I,2)=TST
          PRINT *,' INITIAL PHASE FOR SIGNAL',I,' IN DEGREES(0-360)'
          READ  *,TPH
          WRITE(2,4000)I,TPH
          WRITE(1,4000)I,TPH
4000      FORMAT('INITIAL PHASE FOR SIGNAL',I3,' IS ',F6.2,
     >           ' DEGREES'/)
          TPH=TPH*.01745329252
```

```
            ASIGPAR(I,3)=TPH
            PRINT *,' WAVELENGTH OF SIGNAL ',I,' IN METERS'
            READ  *,WVLTH
            WRITE(2,5000)I,WVLTH
            WRITE(1,5000)I,WVLTH
5000        FORMAT('WAVELENGTH OF SIGNAL',I3,' IS ',F6.2,' METERS'/)
            ASIGPAR(I,4)=WVLTH
10        CONTINUE
          PRINT*
          PRINT *,' THAT COMPLETES THE INPUT SIGNAL PARAMETERS.'
          PRINT *
C
6000      RETURN
C
          END
C
C************************SUBROUTINE INANT*********************************
C
          SUBROUTINE INANT (SEP,ATVEC,NMAT,SIM)
C
          IMPLICIT NONE
C
          INTEGER I,J,K,L,SIM
C
          REAL SEP,ATVEC(160,5),NMAT
C
          PRINT *,' A DESCRIPTION OF THE ANTENNA ARRAY IS NEEDED.'
          PRINT *, ' ANTENNA ARRAY IN LINEAR EQUALLY SPACED'
          PRINT *, ' IDENTICAL RECEIVER ELEMENTS'
          PRINT *, ' HOW MANY ANTENNA ELEMENTS DO YOU WISH ?'
          READ  *,NMAT
C
          PRINT *,' WHAT IS THE SEPARATION OF ANTENNAS IN METERS ?'
          READ  *,SEP
C
          PRINT *,' THAT COMPLETES THE ANTENNA ARRAY DISCRIPTION.'
          PRINT *
C
          RETURN
          END
C
C
C************************SUBROUTINE INNOISE*******************************
C
          SUBROUTINE INNOISE(NMAT,ATVEC,SIM)
C
          IMPLICIT NONE
C
          INTEGER K,SIM
C
          REAL ATVEC(160,5),NMAT,L,M,J,I,TEMP
C
c          IF(SIM.EQ.0) RETURN
          PRINT *, 'EACH ANTENNA/RECEIVER WILL HAVE ITS OWN'
          PRINT *, 'NOISE GENERATORS.  INPUT THE NUMBER OF'
          PRINT *, 'NOISE SOURCES, AND THE DISTRIBUTION OF'
          PRINT *, 'EACH NOISE SOURCE.'
          PRINT *
```

```
      PRINT *, 'IF EACH ANTENNA THE SAME RMS LEVEL AND '
      PRINT *, 'DISTRIBUTION THEN TYPE (1) ELSE (0)'
      PRINT *
C
      READ  *,K
C
      DO 10 I=1,NMAT
         PRINT *,'ANTENNA AT POSITION ',I
         PRINT *,'HOW MANY NOISE SOURCES'
         READ  *,L
         DO 10 M=1,L
            PRINT *,'TYPE OF NOISE GENERATOR NUMBER ',M
            PRINT *,'GAUSSIAN (1)'
            PRINT *,'UNIFORM  (2)'
            PRINT *,'RAYLEIGH (3)'
            PRINT *,'IMPUSE   (4)'
            PRINT *,'BURST    (5)'
            READ  *,J
            PRINT *,'WHAT IS THE RMS NOISE POWER IN MICROWATTS ?'
            READ *,TEMP
            IF (K.EQ.1) THEN
               DO 5 K=1,NMAT
                  ATVEC(K,J)=TEMP
5              CONTINUE
               GOTO 20
            ENDIF
            ATVEC(I,J)=TEMP
10    CONTINUE
C
20    PRINT *,'THAT COMPLETES THE NOISE SOURCES DESCRIPTIONS.'
C
1000  RETURN
C
      END
C
C**********************SUBROUTINE GENOISMTX*****************************
C
      SUBROUTINE GENOISMTX(NMAT,NMSMP,ATVEC,ANOMTX,ISEED,SIM)
C
C THIS PROGRAM GENERATES A UNIFORM NOISE SEQUENCE WITH VALUES IN THE
C RANGE OF 0 TO 1 WITH ZERO MEAN.  USING THIS UNIFORM DISTRIBUTION A
C GUASSIAN NOISE WITH ZERO MEAN AND VARIANCE = 1
C
      IMPLICIT NONE
C
      INTRINSIC LOG10,ABS,SQRT,FLOAT,TAN
C
      EXTERNAL SPRAND2,SPRAND
C
      LOGICAL ACCEPT
C
      INTEGER J,SIM,K,I,ISEED2,ISEED,RECORDS,HIT,NUMOFHITS,L
C
      REAL SPRAND2,SPRAND,SUM,SUM2,RAND,VARIANCE,UNIFORM,
     >  SCALE,MEAN,MAXAMP,MAXAMPINPUT,PI,
     >  POWERS,POWERN,SIGNOFUNIFORM,PERCENT,
     >  NMAT,NMSMP,ATVEC(160,5), ANOMTX(320,320)
C
```

```
      IF(SIM.EQ.0)THEN
         PRINT *,'HOW MANY SAMPLES ARE THERE PER ANTENNA ?'
      ELSE
        PRINT *,'HOW MANY SAMPLES DO YOU WISH TO SIMULATE ?'
      ENDIF
      READ  *,NMSMP
C
      IF(SIM.EQ.0) RETURN
C
C ISEED IS THE INITIAL VALUE (SEED) FOR THE RANDOM NUMBER GENERATOR AND
C SIMPLY MUST BE ANY POSITIVE NONZERO INTEGER
C
      PI          = 22.0/7.0
      ISEED2      = 11357
      POWERS      = 0.0
      POWERN      = 0.0
      MEAN        = 0.0
      MAXAMPINPUT = 0.0
      MAXAMP      = 0.0
      NUMOFHITS   = 3.0
      PERCENT     = 0.03
      HIT         = 0
      VARIANCE    = 1.0
      L           = 0
C     DO 1 I=1,NMAT*2
        DO 1 J=1,NMSMP*2
          ANOMTX(I,J)=0.0
1     CONTINUE
      DO 10 I=1,NMAT*2,2
          L=L+1
          DO 10 K = 1,NMSMP*2,2
            SUM  = 0.0
            SUM2 = 0.0
C IF UNIFORM NOISE
            IF ( ATVEC(L,2).NE. 0 ) THEN
        ANOMTX(I,K) =ATVEC(L,2)* SQRT(12.0*VARIANCE)
     )*(SPRAND(ISEED)-0.5)
        ANOMTX(I,K+1) =ATVEC(L,2)* SQRT(12.0*VARIANCE)
     )*(SPRAND2(ISEED2)-0.5)
        ANOMTX(I+1,K)=-ANOMTX(I,K+1)
        ANOMTX(I+1,K+1)=ANOMTX(I,K)
            ENDIF
C IF GAUSS NOISE
            IF ( ATVEC(L,1) .NE. 0 ) THEN
              DO J = 1,12
                UNIFORM = SPRAND(ISEED)
                SUM=SUM+SQRT(VARIANCE)*(UNIFORM - 0.5)
              ENDDO
              ANOMTX(I,K) =ATVEC(L,1)* SUM
              SUM = 0
C
              DO J = 1,12
              UNIFORM = SPRAND2(ISEED2)
              SUM = SUM + SQRT(VARIANCE)*( UNIFORM - 0.5)
              ENDDO
              ANOMTX(I,K+1) =ATVEC(L,1)* SUM
              ANOMTX(I+1,K)=-ANOMTX(I,K+1)
              ANOMTX(I+1,K+1)=ANOMTX(I,K)
```

```
                ENDIF
C IF RAYLEIGH NOISE
C RAYLEIGH = TRANSFORMATION OF     SQRT ( X1^2 + X2^2 )
C  X1 AND X2 ELEMENTS OF GAUSIAN DIST. (0,1)
                IF ( ATVEC(L,3).NE. 0 ) THEN
                        DO J = 1,12
                        UNIFORM = SPRAND(ISEED)
                        SUM =  SUM  + SQRT(VARIANCE)*( UNIFORM - 0.5)
                        ENDDO
                        DO J = 1,12
                        UNIFORM = SPRAND2(ISEED2)
                        SUM2 = SUM2 + SQRT(VARIANCE)*( UNIFORM - 0.5)
                        ENDDO
                        ANOMTX(I,K)= SUM2/9.0 + 0.5
               ANOMTX(I,K) = ATVEC(L,3)*SQRT( SUM2**2 + SUM**2 )
               ANOMTX(I+1,K+1)=ANOMTX(I,K)
C
               ANOMTX(I,K+1)=ANOMTX(I,K)
               ANOMTX(I+1,K)=-ANOMTX(I,K+1)

                ENDIF
C IF laplacian NOISE
                IF ( ATVEC(L,5).NE. 0 ) THEN
                        UNIFORM = (SPRAND(ISEED)-0.5)
                        SIGNOFUNIFORM = ABS(UNIFORM)/UNIFORM
                        ANOMTX(I,K) = SIGNOFUNIFORM * SQRT(2.0) *
     >                  LOG10(1.0 - 2.0*ABS(UNIFORM) )*ATVEC(L,5)
                        UNIFORM = (SPRAND2(ISEED2)-0.5)
                        SIGNOFUNIFORM = ABS(UNIFORM)/UNIFORM
                        ANOMTX(I,K+1) = SIGNOFUNIFORM * SQRT(2.0) *
     >                  LOG10(1.0 - 2.0*ABS(UNIFORM) )*ATVEC(L,5)
C
                        ANOMTX(I+1,K)=-ANOMTX(I,K+1)
                        ANOMTX(I+1,K+1)=ANOMTX(I,K)
C
                ENDIF
C IF IMPULSE NOISE: PERCENT HITS PERCENT PROBABILITY OF AN IMPULSE
C ASSUMING A UNIFORM DISTRIBUTION
                IF ( ATVEC(L,4).NE. 0 ) THEN
                ANOMTX(I,K) =SQRT(12.0*VARIANCE)*(SPRAND(ISEED)-0.5)
               IF ( ANOMTX(I,K) .GE.  -1.73
     1          .AND. ANOMTX(I,K) .LE. (-1.73 + 2*1.73*PERCENT)
     2          .AND. HIT .LE. NUMOFHITS ) THEN
                                ANOMTX(I,K) =ANOMTX(I,K)+ATVEC(L,4)
                                ANOMTX(I+1,K+1)=ANOMTX(I,K)
                                HIT = HIT + 1
                        ENDIF
                ENDIF

C
10      CONTINUE
C
1000    RETURN
C
        END
C
C************************FUNCTION SPRAND************************************
C
```

```
C FUNCTION TO CALCULATE A RANDOM NUMBER BETWEEN 0 AND 1
C
        FUNCTION SPRAND(ISEED)
C
         ISEED = (2045 * ISEED) + 1
         ISEED = ISEED - (ISEED/1048576)*1048576
         SPRAND = FLOAT(ISEED + 1)/1048577.0
C
        RETURN
C
        END
C
C**********************FUNCTION SPRAND2*********************************
C
C FUNCTION TO CALCULATE A RANDOM NUMBER BETWEEN 0 AND 1
C
        FUNCTION SPRAND2(ISEED2)
C
         ISEED2 = (2045 * ISEED2) + 1
         ISEED2 = ISEED2 - (ISEED2/1048576)*1048576
         SPRAND2 = FLOAT(ISEED2 + 1)/1048577.0
C
        RETURN
C
        END
C
C**********************SUBROUTINE SAMP**********************************
C
      SUBROUTINE SAMP(NMAT,NMSMP,ANOMTX,SMPVEC,SEP,ANMSIG,ASIGPAR,SIM)

        IMPLICIT NONE
C
        EXTERNAL SPRAND2,SPRAND
C
        INTEGER I,J,L,SIM,STARTA,STARTS,ISEED

        REAL SPRAND,ANMSIG,ASIGPAR(159,6),PI,ALPH(159),MAMP(159),
     >       SEP,NSEP,AMP,THETA,ALPHA,A,AR,AI,BR,BI,K,WVLTH,CYCL,
     >       NMAT,NMSMP,ARSGVEC(320,2),SMPVEC(320,320),corel,
     >       ANOMTX(320,320),REALDAT(256,16),IMAGDAT(256,16)
C
C THIS ROUTINE COMPUTES THE SAMPLES OF THE SIGNAL PLUS NOISE
C FOR EACH ANTENNA. THE RESULTS ARE RETURNED IN SMPMTX.
C
         ISEED=12125
1000    FORMAT(16F15.5)
        IF (SIM.EQ.0) THEN
            DO 100 I=1,NMSMP
               READ(3,1000)(REALDAT(I,K),K=1,16)
100         CONTINUE
            CLOSE(3)
            DO 200 I=1,NMSMP
               READ(4,1000)(IMAGDAT(I,K),K=1,16)
200         CONTINUE
            CLOSE(4)
            K=0
            PRINT *,'WHICH SAMPLE SHOULD BE FIRST ?'
            READ *,STARTS
```

```
            DO 5 I=1,NMAT*2,2
               PRINT *,'WHICH ANTENNA SHOULD BE SAMPLED ?'
               READ *,STARTA
               K=STARTA
               L=0
               DO 5 J=1,NMSMP*2,2
                  L=L+STARTS
                  SMPVEC(I,J)=REALDAT(L,K)
                  SMPVEC(I+1,J)=IMAGDAT(L,K)
                  SMPVEC(I,J+1)=-SMPVEC(I+1,J)
                  SMPVEC(I+1,J+1)=SMPVEC(I,J)
5           CONTINUE
         ELSE
C
          print *,'CORELLATION BETWEEN SIGNALS ?, 1 TO 0'
          READ *,COREL
          CYCL=0.0
          PI=3.141592654
        DO 30 L=1,NMSMP*2,2
          CYCL=CYCL+1
          DO 3301 I=1,ANMSIG
             IF (I.EQ.CYCL)THEN
                MAMP(I) = 1.4142135
          ELSE
                MAMP(I) = corel*1.4142135
             ENDIF
             ALPH(I)=SPRAND(ISEED)/10.0
3301          CONTINUE
          IF (CYCL.EQ.ANMSIG)CYCL=0.0
C
          K=0
          DO 20 I=1,NMAT*2,2
                K=K+1
                ARSGVEC(I,1)=0
                ARSGVEC(I,2)=0
                DO 20 J=1,ANMSIG
                   THETA=ASIGPAR(J,1)
                   AMP=ASIGPAR(J,2)
                   ALPHA=ASIGPAR(J,3)
                   WVLTH=ASIGPAR(J,4)
                   A=((((K-((NMAT+1.0)/2.0))*PI*2.0*SEP)/WVLTH)
     >                  *SIN(THETA))
                   AR=COS(A)
                   AI=SIN(A)
                   BR=(AMP*MAMP(J))*COS(ALPHA+ALPH(J))
                   BI=(AMP*MAMP(J))*SIN(ALPHA+ALPH(J))
                   ARSGVEC(I,1)=ARSGVEC(I,1)+(AR*BR-AI*BI)
                   ARSGVEC(I,2)=ARSGVEC(I,2)-(AR*BI+AI*BR)
                   ARSGVEC(I+1,1)=-ARSGVEC(I,2)
                   ARSGVEC(I+1,2)=ARSGVEC(I,1)
20              CONTINUE
          DO 10 I=1,NMAT*2
                SMPVEC(I,L)=ARSGVEC(I,1)+ANOMTX(I,L)
                SMPVEC(I,L+1)=ARSGVEC(I,2)+ANOMTX(I,L+1)
10        CONTINUE
30        CONTINUE
          ENDIF
C
```

```
      RETURN
C
      END
C
C**********************SUBROUTINE INVERT*******************************
C
      SUBROUTINE INVERT (NMAT,B,C,SIM)
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L,SIM
C
      REAL N,A(320,320),B(320,320),C(320,320),NMAT
C
      IF(SIM.EQ.0)THEN
         DO 1010 I=1,NMAT*2
            C(I,I)=1.0
1010     CONTINUE
         RETURN
      ENDIF
C
      N=NMAT*2.0
C
      DO 777 I=1,N
         DO 777 J=1,N
            A(I,J)=B(I,J)
777   CONTINUE
C
C ** MAKE C IDENTITY MATRIX FOR INVERTING
C
      DO 500 I=1,N
500               C(I,I)=1.0
C
C ** DIVIDE EACH ROW BY THE FIRST ELEMENT
C ** ONLY NEED TO DIVIDE C IN LOWER TRIANGLE
C ** DIVIDE FROM RIGHT TO LEFT FOR A SO FIRST
C ** ELEMENT DOES NOT GO TO ONE BEFORE USE
C
      DO 10  L=0,N-1
C
         DO 20 I=1+L,N
C
C ** CHECK TO SEE IF ALREADY ONE, IF SO NO DIVIDE
C ** IS NECESSARY FOR THAT ROW, ALSO IF AREADY ZERO
C
            IF(A(I,L+1).EQ.1) GOTO 20
C
            IF(A(I,L+1).EQ.0) GOTO 20
C
            DO 200 J=1,I
C
200            C(I,J)=C(I,J)/A(I,L+1)
C
            DO 201 J=0,N-1
C
201            A(I,N-J)=A(I,N-J)/A(I,L+1)
C
20       CONTINUE
```

```
C
C ** YOU NOW HAVE THE FIRST COLUMN AS ALL ONES, THE
C ** NEXT STEP IS TO SUBTRACT THE TOP ROW  FROM THE
C ** ROW BELOW IN TOP TO BOTTOM, LEFT TO RIGHT  ORDER.
C
           DO 30 I=2+L,N
C
              IF(A(I,1+L).EQ.0)GOTO 30
C
              DO 300 J=1,I
C
300              C(I,J)=C(I,J)-C(L+1,J)
C
              DO 301 J=2+L,N
C
301              A(I,J)=A(I,J)-A(L+1,J)
C
30         CONTINUE
C
10      CONTINUE
C
C ** AT THIS POINT THE A MATRIX THAT IS LEFT
C ** IS UPPER TRIANGULAR AND THE C MATRIX IS LOWER
C ** TRIANGULAR. THIS IS A LU DECOMPOSITION
C ** OF THE ORINGINAL A MATRIX. THE ONES WERE NOT
C ** SUBTRACTED AND MUST BE SET TO ZERO BEFORE
C ** THE NEXT STEP IS BEGUN.
C
        DO 90 I=2,N
C
           DO 90 J=1,I
C
90            A(I,J)=0
C
C ** AT THIS POINT THE A MATRIX IS JUST AN UPPER TRIANGULAR
C ** MATRIX THAT HOLDS THE MULTIPLIER FOR THE C MATRIX.  THE
C ** PROCEDURE APPLIED HERE IS TO MULT AND SUBTRACT TO REDUCE
C ** THE A MATRIX TO THE ROW REDUCED ECHELON FORM, LEAVING THE
C ** C MATRIX AS THE INVERTED A MATRIX.  ONLY THE LOWER HALF IS
C ** ACTUALLY COMPUTED, THE UPPER HALF WILL BE COPIED INTO THE
C ** UPPER HALF TO SAVE COMPUTATIONS IN LARGE MATRICES.
C
        DO 60 L=0,N-1
C
           DO 60 I=1,N-1-L
C
              DO 60 J=1,I
C
60               C(I,J)=C(I,J)-C(N-L,J)*A(I,N-L)
C
       DO 80 I=1,N-1
          DO 80 J=I+1,N
80           C(I,J)=C(J,I)
C
        RETURN
C
        END
C
```

```
C************************SUBROUTINE CREATEFILE*****************************
C
      SUBROUTINE CREATEFILE(NMAT,NMSMP,ATVEC,ANOMTX,AENOMTX,AIENOMTX,
     >                      SMPVEC,SEP,ARSGVEC,SIM)
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L,SIM
C
      REAL NMAT,NMSMP,ANOMTX(320,320),AENOMTX(320,320),
     >     AIENOMTX(320,320),SMPVEC(320,320),ATVEC(160,5),
     >     SEP,ARSGVEC(320,2)
C
      WRITE(2,1001)NMAT,NMSMP,SEP
      WRITE(1,1001)NMAT,NMSMP,SEP
1001  FORMAT(X,'NUMBER OF ANT= ',F3.0,/' NUMBER OF SAMPLES=',F4.0,
     >       /' SEPARATION BETWEEN ELEMENTS= ',F8.5,' METERS')
C
      WRITE(1,1012)
      DO 10 I=1,NMAT*2
         DO 10 J=1,NMAT*2
            WRITE(1,1002)I,J,AENOMTX(I,J),AIENOMTX(I,J)
10    CONTINUE
C
1012  FORMAT(/'   I  ',' J  ',' AENOMTX(I,J)',' AIENOMTX(I,J)'/)
1002  FORMAT(I4,I4,2X,F10.7,4X,F10.7)
C
      WRITE(1,1013)
1013  FORMAT(/'  SAMPLE #  ','ANOMTX(I,J) VALUE'/)
1003  FORMAT(I4,2X,E17.7)
      DO 20 I=1,NMAT*2,2
         WRITE(1,1113)I/2+1
1113  FORMAT('ANTENNA NUMBER ',I3)
         DO 20 J=1,NMSMP*2,2
            WRITE(1,1003)J,ANOMTX(I,J)
            WRITE(1,1003)J+1,-ANOMTX(I,J+1)
20    CONTINUE
C
      WRITE (1,1014)
      DO 30 I=1,NMAT
         DO 30 J=1,5
            WRITE(1,1004)I,J,ATVEC(I,J)
30    CONTINUE
1004  FORMAT(I4,I4,3X,F10.5)
1014  FORMAT(/'   I ',' J  ', '     ATVEC(I,J)'/)
C
      WRITE(1,1015)
      DO 40 I=1,NMAT*2
         DO 40 J=1,2
            WRITE(1,1005)I,J,ARSGVEC(I,J)
40    CONTINUE
1005  FORMAT(I4,I4,3X,F10.3)
1015  FORMAT(/'   I ',' J  ',' ARSGVEC(I,J)'/)
C
      RETURN
C
      END
C
```

```
C************************SUBROUTINE NOEXPECT*********************************
C
      SUBROUTINE NOEXPECT(NMAT,NMSMP,SMPVEC,ESMPMTX,SIM)
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L,SIM
C
      REAL NMAT,SMPVEC(320,320),ESMPMTX(320,320),VEC(320,2),
     >     NMSMP,MTX(320,320),NORM,NORM2
C
C  THIS ROUTINE CONVERTS THE NOISE SAMPLE MATRIX INTO NMATXNMAT
C  MATRIX THAT REPRESENTS THE EXPECTED VALUE
C  OF THE SAMPLES ALLOWING A USED IN THE GENERALIZED EIG PROCESS TO
C  IMPROVE THE ESTIMATE.
C
      IF(SIM.EQ.0)THEN
        DO 1010 I=1,NMAT*2
           ESMPMTX(I,I)=1.0
1010    CONTINUE
        RETURN
      ENDIF
C
      DO 10 J=1,NMSMP*2,2
         DO 20 I=1,NMAT*2
            VEC(I,1)=SMPVEC(I,J)
            VEC(I,2)=SMPVEC(I,J+1)
20       CONTINUE
C
            CALL VCCNJMP(NMAT,VEC,MTX)
C
            DO 10 K=1,NMAT*2
               DO 10 L=K,NMAT*2
                  ESMPMTX(L,K)=ESMPMTX(L,K)+MTX(L,K)
10    CONTINUE
C
         DO 45 I=1,NMAT*2
            DO 45 J=I,NMAT*2
               ESMPMTX(I,J)=ESMPMTX(J,I)
45       CONTINUE
C
         DO 50 I=1,NMAT*2,2
            NORM=NORM+ESMPMTX(I,I)
50       CONTINUE
C
         NORM=NMAT/NORM
         DO 60 I=1,NMAT*2
            DO 60 J=1,NMAT*2
               ESMPMTX(I,J)=ESMPMTX(I,J)*NORM
60       CONTINUE
C
      RETURN
C
      END
C
C************************SUBROUTINE VECNORM*********************************
C
      SUBROUTINE VECNORM(NMAT,VEC,NORM2)
```

```
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L
C
      REAL NMAT,VEC(320,2),NORM2
C
      NORM2=0.0
C
      DO 10 I=1,NMAT*2
         NORM2=NORM2+VEC(I,1)**2+VEC(I,2)**2
10    CONTINUE
C
      RETURN
C
      END
C
C**********************SUBROUTINE VCCNJMP*********************************
C
      SUBROUTINE VCCNJMP(NMAT,VEC,MTX)
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L
C
      REAL NMAT,VEC(320,2),MTX(320,320)
C
      DO 10 J=1,NMAT*2
         DO 10 I=J,NMAT*2
            MTX(I,J)=VEC(I,1)*VEC(J,1)+VEC(I,2)*VEC(J,2)
10    CONTINUE
C
      RETURN
C
      END
C
C**********************SUBROUTINE ESNOISE*****************************
C
      SUBROUTINE ESNOISE(NMAT,NMSMP,ATVEC,ANOMTX,ISEED2,SIM)
C
      IMPLICIT NONE
C
      INTEGER I,J,K,L,SIM,ISEED2
C
      REAL ATVEC(160,5),ANOMTX(320,320),NMAT,NMSMP,A
C
C THIS ROUTINE ESTIMATES THE NOISE WITHOUT APRIORI INFORMATION. IT
C PROVIDES AN INVERTED CORRELATION MATRIX OF THE ESTIMATE AS EINOMTX
C AS THE OUTPUT FOR USE IN THE SOLUTION OF THE GENERALIZED EIGENVALUE
C PROBLEM.
C
      PRINT *,'USE PERFECTLY COREL(0), ESTIMATED( 1), OR ACTUAL(2)'
C
      READ *,A
C
      IF(A.EQ.2)THEN
         SIM=1
         RETURN
```

```
      ENDIF
      IF(A.EQ.1)THEN
         SIM=1
         CALL GENOISMTX(NMAT,NMSMP,ATVEC,ANOMTX,ISEED2,SIM)
         RETURN
      ENDIF
      IF(A.EQ.0)THEN
        SIM=0
      ENDIF
C
      RETURN
C
      END
C
C***  ALL NODE WORK FROM HERE ***** THIS IS PARALLEL DOA ************
C
      SUBROUTINE EIGPOWER(NODES,WVLTH,SEP,NMSMP,SMPVEC,EINOMTX,
     >                    ENOMTX,NMAT,OUTMSG,SIM)
      IMPLICIT NONE
C
      REAL B(320,320),C(320,320),X(320,320),Y(320),
     >     EV,VCK,MAX(320),S,TOL,D,EIGVEC(320,2),ESMPMTX(320,320),
     >     V,N,SKP,MYA,TIMES,NCPN,EINOMTX(320,320),NMAT,GEN,
     >     ENOMTX(320,320),SIZE(10),SIZE1,EIGENSYS(325),NODES,
     >     PARAMSG(102400),SECDMSG(102400),THRDMSG(102400),STARTMSG,
     >     SMPVEC(320,320),NMSMP,SP,WLTH,SEP,WVLTH,PSZ,ZPZ,
     >     OUTMSG(1834),DOA,MAXAMP,ANGLE,START,INC,TIME1,IZE
C
      INTEGER  K,L,I,J,R,SIM,
     >         SMS, EMS, TMS, MS, TSEC, SEC, MIN, ALLNODES,
     >         CUBETYPE,PARAMTYPE,INITTYPE,EIGTYPE,SECDTYPE,THRDTYPE,
     >         CUBESIZE,PARAMSIZE,INITSIZE,EIGSIZE,OUTSIZE,OUTTYPE,
     >         ROOT, HOST1, HOSTPID,ROOTNODE,APPLPID,STARTTYPE,
     >         WORKNODES,MYNOD,PID,NEW,NUM
C

C  MESSAGE AREAS DEFINED
C
C     MAKE EXPLICIT THE STRUCTURE OF EIGENSYS:
C
      EQUIVALENCE (EIGENSYS(1), EV)
      EQUIVALENCE (EIGENSYS(2), SKP)
      EQUIVALENCE (EIGENSYS(3), MYA)
      EQUIVALENCE (EIGENSYS(5), TIME1)
      EQUIVALENCE (EIGENSYS(6), Y(1))
C
C     MAKE EXPLICIT THE STRUCTURE OF SIZE:
C
      EQUIVALENCE (SIZE(1),  TIMES)
      EQUIVALENCE (SIZE(2),  N)
      EQUIVALENCE (SIZE(3),  TOL)
      EQUIVALENCE (SIZE(4),  SP)
      EQUIVALENCE (SIZE(5),  GEN)
      EQUIVALENCE (SIZE(6),  WLTH)
      EQUIVALENCE (SIZE(7),  S)
      EQUIVALENCE (SIZE(8),  PSZ)
      EQUIVALENCE (SIZE(9),  IZE)
      EQUIVALENCE (SIZE(10), ZPZ)
```

```
C
C       MAKE EXPLICT THE STRUCTURE OF PARAMSG
C
        EQUIVALENCE (PARAMSG(1), X(1,1))
C
C       MAKE EXPLICT THE EQUIVALENCE OF B,C AND SECD AND THRDMSG
C
        EQUIVALENCE (SECDMSG(1), B(1,1))
        EQUIVALENCE (THRDMSG(1), C(1,1))
C
      DATA CUBETYPE /0/,PARAMTYPE /1/,INITTYPE  /5/,EIGTYPE /30/,
     >      SECDTYPE/2/,THRDTYPE/3/,OUTTYPE/35/,STARTTYPE/100/,
     >      CUBESIZE/40/ ,PARAMSIZE /409600/,OUTSIZE/7336/,
     >      ROOT /-32768/,ROOTNODE /0/, HOSTPID /100/, APPLPID /0/,
     >      ALLNODES /-1/
C
        PRINT *,'MAX EST(1), TRACE EST(2), MIN EST (3)'
        READ *,I
        IF (I.EQ.2) THEN
        CALL LOAD ('node',-1,0)
        ELSE
        IF (I.EQ.3) THEN
        CALL LOAD ('mineig',-1,0)
        ELSE
        CALL LOAD ('maxeig',-1,0)
        ENDIF
        ENDIF
        PRINT *,'LOADING THE NODES ...'
C
        GEN=REAL(SIM)
        IZE=2**NODES
        SP=SEP
        print*,'WHAT WAVELENGTH DO YOU WISH TO DETECT ?',WVLTH
        READ *,WVLTH
        WLTH=WVLTH
        SIZE1=IZE
        S=INT(2*NMSMP/IZE)
        N=2*NMAT
        EIGSIZE=INT(4*(N+5))
        PRINT *,' ESTIMATE NUMBER OF SIGNALS'
        READ *,TIMES
        PRINT *, 'WHAT TOL?'
        READ *,TOL
        PSZ=320*S*4
        R=MOD(2*NMSMP,IZE)
        ZPZ=320*S*4+4*R*320
        CALL FLUSHMSG(-1,-1,-1)
        DO 1000 I=1,N
            DO 1000 J=1,N
               B(I,J)=EINOMTX(I,J)
               C(I,J)=ENOMTX(I,J)
1000        CONTINUE
        DO 3333 I=1,N
           DO 3333 J=1,NMSMP*2
              X(I,J)=SMPVEC(I,J)
3333    CONTINUE
C SEND SIZE TO USE INTO CUBE
C
```

```
        IZE=NODES
        CALL CSEND(CUBETYPE,SIZE, CUBESIZE,ALLNODES,APPLPID)
C
C SEND PARAMETERS INTO CUBE
C
        CLOSE(1)
C
        IZE=2**NODES
        PARAMSIZE=INT(PSZ)
        NCPN=INT(N/IZE)
        INITSIZE=INT(4*(NCPN*320))
        DO 66 I=1,IZE-1
             J=((I-1)*(S*320)+1)
             K=((I-1)*(NCPN*320)+1)
             CALL CSEND(PARAMTYPE,PARAMSG(J),PARAMSIZE,I,APPLPID)
             CALL CSEND(SECDTYPE,SECDMSG(K),INITSIZE,I,APPLPID)
             CALL CSEND(THRDTYPE,THRDMSG(K),INITSIZE,I,APPLPID)
66      CONTINUE
C
        PARAMSIZE=INT(ZPZ)
        K=(S*(IZE-1)*320)+1
        CALL CSEND(PARAMTYPE,PARAMSG(K),PARAMSIZE,0,APPLPID)
        R=MOD(N,IZE)
        INITSIZE=INT(4*(NCPN*320)+4*R*320)
        K=(NCPN*(IZE-1)*320)+1
        CALL CSEND(SECDTYPE,SECDMSG(K),INITSIZE,0,APPLPID)
        CALL CSEND(THRDTYPE,THRDMSG(K),INITSIZE,0,APPLPID)
C
        PRINT *,'NODES LOADED READY TO START, TYPE (1)'
        READ *,STARTMSG
        CALL CSEND(STARTTYPE,STARTMSG,4,-1,0)
C
C       RECEIVE FINAL EGENSET VALUES FROM CUBE, AND THE TIME,
C       IN MILLISECONDS (AS 'SKP')
C
        PRINT *,'HOST WAITING TO RECEIVE EIGENSET VALUES FROM CUBE'
C
        CALL CRECV(EIGTYPE, EIGENSYS, EIGSIZE)
C
        print*,'EIGENVALUE=',EV
        WRITE(2,400) EV
400     FORMAT (/' EIGENVALUE=',F25.7/)
C
        PRINT *,'ACTUAL ITERATIONS=',EIGENSYS(4),MYA
        WRITE(2,510)EIGENSYS(4),MYA
510     FORMAT(/'ACTUAL ITERATIONS=',F4.0,' and ',F4.0/)
C
        WRITE(2,500)
500     FORMAT(/' ASSOCIATED EIGENVECTOR=',F25.7/)
        DO 20 J=1,N
           WRITE(2,600) J,Y(J)
600        FORMAT(' X(',I5,')=',F20.7)
20      CONTINUE
C
        TMS =NINT(TIME1)
        MS  = MOD(TMS, 1000)
        TSEC = (TMS - MS) / 1000
        SEC  = MOD(TSEC, 60)
```

```
        MIN  = (TSEC - SEC) / 60
C
        WRITE(6, 111) MIN, SEC, MS
C
        WRITE (2,111) MIN,SEC, MS
C
        TMS =NINT(SKP)
        MS   = MOD(TMS, 1000)
        TSEC = (TMS - MS) / 1000
        SEC  = MOD(TSEC, 60)
        MIN  = (TSEC - SEC) / 60
C
        WRITE(6, 111) MIN, SEC, MS
C
        WRITE (2,111) MIN,SEC, MS
C
111     FORMAT('  ELAPSED TIME = ',I2,' MIN. ',I2,'.',I3.3, ' SEC.')
C
        CALL CRECV(OUTTYPE,OUTMSG,OUTSIZE)
C
        TMS =NINT(OUTMSG(1812))
        MS   = MOD(TMS, 1000)
        TSEC = (TMS - MS) / 1000
        SEC  = MOD(TSEC, 60)
        MIN  = (TSEC - SEC) / 60
        WRITE (2,111) MIN,SEC, MS
        WRITE(6, 111) MIN, SEC, MS
C
        DO 1813 I=1813,1822
           WRITE(2,123)OUTMSG(I+11),OUTMSG(I)
           WRITE(6,123)OUTMSG(I+11),OUTMSG(I)
1813    CONTINUE
123     FORMAT(' DOA=',F8.2,'MAGNITUDE=',F20.5)
C
C       CLEAN OUT UNRECEIVED MESSAGES
C
C
C
C********END OF NODE PARALLEL PROCESSING **** ONLY OUTPUT LEFT*********
C
            RETURN
C
            END
C
C************************SUBROUTINE OUT************************************
C
C
        SUBROUTINE OUT(OUTMSG)
C
        IMPLICIT NONE
C
        INTEGER I,J,K,L
C
        REAL OUTMSG(1834),ANGLE
C
C  OUTPUTS THE VALUES OF THE ESITMATED DIRECTIONS OF ARRIVAL,
C
        ANGLE=-90.1
```

```
      DO 200 I=1,1801
         ANGLE=ANGLE+.1
            J=NINT(10*ANGLE)
            IF((MOD(J,10).EQ.0).OR.(OUTMSG(I).GT.1000))THEN
            IF((ANGLE.GT.89.9).OR.(ANGLE.LT.-89.9))OUTMSG(I)=1.0
            IF(OUTMSG(I).LT..00001)OUTMSG(I)=.00001
            WRITE (15,100)ANGLE,OUTMSG(I)
         ENDIF
200   CONTINUE
100   FORMAT (F10.2,F20.5)
C
      CALL KILLCUBE(-1,-1)
C
      RETURN
C
      END
C
```

APPENDIX B

NODE FORTRAN COMPUTER PROGRAM

This appendix provides the FORTRAN-386 computer programs that are the node programs. The primary functions of the node program are to compute the four tasks identified in the thesis necessary for the DOA solution. The data is input to the nodes from the host program EIGPOWER and the data of the solution is returned to EIGPOWER for output to the user. Node zero has the additional task of executive node, as well as sharing the lower level work of computation of the operations. The other nodes also have different requirements during the processing, but the same program is used in each with the differences being based on the node number.

Since the four operations are clearly marked in the programs as well as having a reasonable set of comments, this discussion will not address the processing that is the implementation of the thesis in any detail. Only a short functional overview will be presented.

As there are four tasks, there are four working parts to the node program. Node zero acts as the executive, but is also used as a computational node. The workload is evenly distributed for a static workload balance situation.

The first operation computes the sample covariance matrix from the snapshots of data. The wrapping and unwrapping referred to in the comments allows for shorter message traffic between nodes.

The second operation is the parallel power method for the

eigen-decomposition. The results desired are the maximum and minimum eigenvectors. This is also the section where the minimum number of iterations are related to the number of arriving wavefronts. Although it can be automated, it is manually completed in this version of the code. This version will accept an estimate of the number of arriving waves, and then shift the matrix based on that estimate. The reverse process is necessary as described in the dissertation when the number of arriving waves are unknown.

The third and fourth operations are combined as was discussed in the dissertation. It is a straight forward computation of the DOA amplitudes based on sweeping theta through 180/P degrees in one tenth of a degree increments.

The next discussion will briefly describe some the unique parallel processor iPSC/2 FORTRAN-386 routines that are contained in the node programs. This information was extracted from the preliminary iPSC/2 Green Hills FORTRAN Language Reference Manual and is provided only for general reference and understanding to the code in this appendix.

MYHOST returns the node id of the caller's host machine for use in send and receive subroutine calls.

MYPID returns the process id of the calling process. This is the process id that was supplied from the host when the process was loaded.

MYNODE returns the node id of the calling process.

MYCLOCK routine provides a simple mechanism to measure time intervals.

CRECV initiates the receipt of a message. The CRECV call waits for a message whose TYPE matches the TYPE specified. When the message is received, it is stored in the buffer specified, and the calling process resumes execution. The CRECV subroutine is synchronous, causing the calling process to be blocked until the desired message is received.

CSEND is used to send a message to a node or host process. The

completion of the CSEND does not imply that the message was received by the destination process, only that the message was sent and that the buffer is available for reuse. CSEND is synchronous, causing the calling process to be blocked until the send operation is complete.

```
      PROGRAM nodE
C
      IMPLICIT NONE
C
      EXTERNAL MYHOST,MYPID,MYNODE,MCLOCK
C
      REAL A(320,320),B(320,320),C(320,320),XD(320,320),AS(102400),
     >     X(320),Y2(320),Y(320),Z(320,320),MAX(6),MY(320),OADOA(11),
     >     TS,S,TOL,V,TEV,EV,VCK,TMYA,AMAXAMP,MAXAMP,ADOA(11),DOA(11),
     >     N,MYA,MYB,SKP,TIMES,D,TSKP,NXTEV,NXTS,NCOL,XDS(102400),
     >     RALY,RAL,AATVAL(160,2),EIGVEC(320,2),TIME1,MAXVEC(320,2),
     >     PLOT(1810),PI,ALPHA,L,T,T1,WVLTH,NMAT,NSEP,SEP,AG,GEN,PK,
     >     PEAK(1810),SSTART,START,FINISH,INC,NUM,ANGLE,NMSIG,DOWN,
     >     VAL1,VAL2,K2,J2,L1,L2,M,PSZ,ZPZ,IZE,ODOA(11),OSTART,
     >     OMAXAMP,J3,J4

C  MESSAGE AREAS DEFINED
C
      REAL SIZE(10),EIGENSYS(325),ITERMSG(325),PARAMSG(102400),plug,
     >     SECDMSG(102400),THRDMSG(102400),SECOND,NMSP,OUTMSG(1834),
     >     COVMSG(102400),COVUPMSG(102400),STARTMSG,OOUTMSG(1834)
C
      INTEGER I,J,K,ALLNODES,R,NCPN,EXT,FCOVTYPE,FCOVSIZE,YSIZE,
     >        CUBETYPE,PARAMTYPE,INITTYPE,ITERTYPE,EIGTYPE,OUTTYPE,
     >        CUBESIZE,PARAMSIZE,INITSIZE,ITERSIZE,EIGSIZE,OUTSIZE,
     >        COVTYPE,COVSIZE,ROOT, HOST, HOSTPID, ROOTNODE,APPLPID,
     >        MYNOD, PID,STARTTIME,DIM,DEST,SECDTYPE,THRDTYPE,YTYPE,
     >        STARTTYPE,SCOVSIZE
C
C     MAKE EXPLICIT THE STRUCTURE OF SIZE:
C
      EQUIVALENCE (SIZE(1),   NMSIG)
      EQUIVALENCE (SIZE(2),   N)
      EQUIVALENCE (SIZE(3),   plug)
      EQUIVALENCE (SIZE(4),   SEP)
      EQUIVALENCE (SIZE(5),   GEN)
      EQUIVALENCE (SIZE(6),   WVLTH)
      EQUIVALENCE (SIZE(7),   NMSP)
      EQUIVALENCE (SIZE(8),   PSZ)
      EQUIVALENCE (SIZE(9),   IZE)
      EQUIVALENCE (SIZE(10),  ZPZ)
C
C     MAKE EXPLICIT THE STRUCTURE OF PARAMSG
C
      EQUIVALENCE (PARAMSG(1),  Z(1,1))
C
C     MAKE EXPLICIT THE EQUIV OF COV,SEC,THRD,COVUPMSG
C
      EQUIVALENCE (COVMSG(1),    A(1,1))
      EQUIVALENCE (SECDMSG(1),   B(1,1))
      EQUIVALENCE (THRDMSG(1),   C(1,1))
      EQUIVALENCE (COVUPMSG(1), XD(1,1))
C
C     MAKE EXPLICIT THE STRUCTURE OF EIGENSYS:
C
      EQUIVALENCE (EIGENSYS(1),  EV)
      EQUIVALENCE (EIGENSYS(2),  SKP)
      EQUIVALENCE (EIGENSYS(3),  TMYA)
```

```
      EQUIVALENCE (EIGENSYS(4),  RAL)
      EQUIVALENCE (EIGENSYS(6),  Y(1))
C
C     MAKE EXPLICIT THE STRUCTURE OF ITERMSG:
C
      EQUIVALENCE (ITERMSG(1),   NXTEV)
      EQUIVALENCE (ITERMSG(2),   NXTS)
      EQUIVALENCE (ITERMSG(3),   NCOL)
      EQUIVALENCE (ITERMSG(4),   SECOND)
      EQUIVALENCE (ITERMSG(5),   S)
      EQUIVALENCE (ITERMSG(6),   X(1))
C
C     MAKE EXPLICIT THE STRUCTURE OF ITERMSG
C
      EQUIVALENCE (OUTMSG(1),      PLOT(1))
      EQUIVALENCE (OUTMSG(1812),   START)
      EQUIVALENCE (OUTMSG(1813),   DOA(1))
      EQUIVALENCE (OUTMSG(1824),   ADOA(1))
C

      EQUIVALENCE (OOUTMSG(1),      PEAK(1))
      EQUIVALENCE (OOUTMSG(1812),   OSTART)
      EQUIVALENCE (OOUTMSG(1813),   ODOA(1))
      EQUIVALENCE (OOUTMSG(1824),   OADOA(1))
C
      EQUIVALENCE (XDS(1), XD(1,1))
      EQUIVALENCE (AS(1),  A(1,1))
C
     DATA CUBETYPE /0/,PARAMTYPE /1/,INITTYPE /5/,OUTTYPE/35/,
     >     CUBESIZE/40/ ,SECDTYPE/2/,THRDTYPE/3/,YSIZE/1280/,
     >     ITERTYPE /15/,EIGTYPE /20/,COVTYPE/25/,OUTSIZE/7336/,
     >     ROOT /-32768/,ALLNODES /-1/,COVSIZE/103040/,FCOVTYPE/40/,
     >     ROOTNODE /0/, HOSTPID /100/, APPLPID /0/,STARTTYPE/100/
      DATA X/320*1.0/
C
      HOST   = MYHOST()
      PID    = MYPID()
      MYNOD = MYNODE()
      NXTEV = 1.0
      SECOND= 0.0
      TS=0
C
C     GET THE SIZE OF THE CUBE TO PUT TO WORK ON PROBLEM
C
      CALL CRECV(CUBETYPE, SIZE, CUBESIZE)
C
      NMSIG=NINT(NMSIG)+1
      TIMES=200.0
      NMAT=N/2.0
      SCOVSIZE=NMAT**2+NMAT
      COVSIZE=SCOVSIZE*4
      YSIZE=N*4
      DIM=IZE
      IZE=2**IZE
      NCPN=INT(N/IZE)
      I=INT(N)
      J=INT(IZE)
      EXT=MOD(I,J)
```

```
      ITERSIZE=(4*(N+5))
      EIGSIZE=(4*(N+5))
      TSKP=0
      SKP=0
C
C     NODES LABLED 0 TO 31
      IF (MYNOD.GE.IZE) GOTO 100
C
      IF (MYNOD.GT.0) GOTO 99
C
C*****************************************************************
C THIS IS THE NODE 0 PROGRAM ONLY
C*****************************************************************
C
      MYA=NCPN*(IZE-1.0)+1.0
C
      MYB=N
C
      PARAMSIZE=ZPZ
      CALL CRECV(PARAMTYPE,PARAMSG(1),PARAMSIZE)
C
      INITSIZE=4*(NCPN*320)+4*EXT*320
C
      CALL CRECV(SECDTYPE, SECDMSG(1), INITSIZE)
C
      CALL CRECV(THRDTYPE, THRDMSG(1), INITSIZE)
C
      CALL CRECV(STARTTYPE,STARTMSG,4)
C
C  START THE CLOCK
C
      STARTTIME=MCLOCK()
C
C*******    TASK ONE    ***    TASK ONE ***    TASK ONE ******************
C
C  THIS ROUTINE CONVERTS THE SAMPLE MATRIX INTO A NXN
C  MATRIX THAT ESTIMATES THE EXPECTED VALUE
C  OF THE SAMPLES ALLOWING A TEMORAL AVERAGING PROCESS TO
C  IMPROVE THE SIGNAL TO NOISE RATIO
C
      DO 3410 J=1,NMSP,4
         J2=J+1
         J3=J2+1
         J4=J3+1
         DO 3410 K=1,N,2
            DO 3410 L=K,N
               VAL1=Z(K,J)*Z(L,J)+Z(K,J3)*Z(L,J3)
               VAL2=Z(K,J2)*Z(L,J2)+Z(K,J4)*Z(L,J4)
               A(K,L)=A(K,L)+VAL1+VAL2
3410     CONTINUE
C WRAPPING UP TO RECEIVE AND ADD TO MATRICES COMPUTED HERE
         K=0
         INC=(2*320+2)
         START=-INC+1
         FINISH=N*320
         DO 9342 J=1,NMAT
            START=START+INC
            DO 9342 I=START,FINISH,320
```

```
                K=K+1
                AS(K)=AS(I)
9342        CONTINUE
            DO 5885 J=1,DIM
               CALL CRECV(COVTYPE,COVUPMSG(1),COVSIZE)
               DO 5885 I=1,SCOVSIZE
               AS(I)=AS(I)+XDS(I)
5885        CONTINUE
C  UNWRAPPING PROCEDURE TO UNSTRING VARIABLE
           L2=320*N-1+2+N
            K=SCOVSIZE+1
           DO 3333 I=1,NMAT
              L2=L2-2
              L=L2
              DO 3333 J=1,2*I
                 K=K-1
                 L=L-320
                 AS(L)=AS(K)
3333        CONTINUE
C
           DO 3440 K=3,N-1,2
              K2=K+1
              DO 3440 L=1,K-2,2
                 L2=L+1
                 A(K,L)=A(L,K)
                 A(K,L2)=-A(L,K2)
3440        CONTINUE
C
           DO 3450 I=1,DIM
           FCOVSIZE=320*NCPN*4*(2**(DIM-I))
           L=(NCPN*(2**(DIM-I))*320)+1-(NCPN*320)
           DEST=(2**(DIM-I))
           CALL CSEND(FCOVTYPE,COVMSG(L),FCOVSIZE,DEST,0)
3450        CONTINUE
C
           DO 6789 I=1,N,2
              TS=TS+A(I,I)
6789        CONTINUE
           TS=TS/(NMSIG-1)
C
           L=(NCPN)*(IZE-1)
           DO 3553 I=1,N
              K=L
              DO 3553 J=1,NCPN+EXT
                 K=K+1
                 A(I,J)=A(I,K)
3553        CONTINUE
C
           DO 3430 K=1,N-1,2
              K2=K+1
              DO 3430 L=1,NCPN+EXT-1,2
                 L2=L+1
                 A(K2,L)=-A(K,L2)
                 A(K2,L2)=A(K,L)
3430         CONTINUE
C
C
C**********   TASK 2   ***   TASK 2   ***   TASK 2   ******************
```

```
C
C  ROUTINE EIGPOWER TAKES THE A MATRIX, WHICH MAY BE SHIFTED
C  AND OUTPUTS THE ESTIMATED
C  EIGENVECTOR, AND EIGENVALUE.
C
        D   =0.0
        SKP =0.0
        TSKP=0.0
C
        TIME1=MCLOCK()-STARTTIME
C
9201    DO 108 L=1,TIMES
           IF((S.NE.0.0).AND.(D.EQ.0.0))THEN
              DO 810 J=1,N
                   DO 810 I=1,NCPN+EXT
                      A(J,I)=A(J,I)-S*C(J,I)
810             CONTINUE
           D=1
           ENDIF
           IF(GEN.EQ.0.0)THEN
             K=0
             DO 1876 I=MYA,MYB
                Y(I)=0
                K=K+1
                DO 1876 J=1, N
                   Y(I)=Y(I)+X(J)*A(J,K)
1876            CONTINUE
            ELSE
               K=0
               DO 789 I=1,N
                  Y2(I)=0
789             CONTINUE
               DO 876 I=MYA,MYB
                  K=K+1
                  DO 876 J=1, N
                     Y2(I)=Y2(I)+X(J)*A(J,K)
876            CONTINUE
               IF (DIM.GT.0.0)THEN
               DO  2330 J=1,DIM
                   YTYPE=2**(J-1)
                   CALL CRECV(YTYPE,MY,YSIZE)
                   DO 2330 I=1,N
                      Y2(I)=Y2(I)+MY(I)
2330           CONTINUE
               CALL CSEND (99,Y2,YSIZE,-1,APPLPID)
               ENDIF
               K=0
               DO 886 I=MYA,MYB
                  Y(I)=0.0
                  K=K+1
                  DO 886 J=1,N
                     Y(I)=Y(I)+Y2(J)*B(J,K)
886            CONTINUE
            ENDIF
C
C  LOOK FOR LARGEST VALUE TO NORMALIZE VECTOR.  THIS WILL
C  BE THE ESTIMATE FOR THE EIGENVALUE.
C
```

```
                X(MYA)=Y(MYA)
                TEV=X(MYA)
                DO 830 J=MYA+1, MYB
                   X(J)=Y(J)
                   IF(ABS(TEV).LT.ABS(X(J)))TEV=X(J)
830             CONTINUE
C
C*******************************************************
C
         DO 88 J=1,DIM
C
                EIGTYPE=2**(J-1)
                CALL CRECV(EIGTYPE,EIGENSYS,EIGSIZE)
C
                IF(ABS(TEV).LT.ABS(EV)) TEV=EV
                DO 88 I=TMYA,TMYA+((2**(J-1))*NCPN)-1
88                    X(I)=Y(I)
C
          DO 11 J=1,N
             X(J)=X(J)/TEV
11        CONTINUE
C
         TOL=ABS(TEV*.0001)
         IF(ABS(NXTEV-TEV).GT.TOL)TSKP=1
         NXTEV=TEV
         IF((TSKP.EQ.0).or.(I.eq.plug)) GOTO 125
C
2201     IF(DIM.GT.0)CALL CSEND(ITERTYPE,ITERMSG,ITERSIZE,-1,0)
C
          IF(SECOND.EQ.1)THEN
          DO 641 I=1,N
             X(I)=1.0
641       CONTINUE
          SECOND=0.0
          TSKP=0.0
          GOTO 8201
          ENDIF
          IF(SECOND.EQ.2)THEN
          GOTO 4004
          ENDIF
          TSKP=0.0
108       CONTINUE
C
125       IF(S.EQ.0)THEN
          S=TS
          TIMES=200.0
          K=0.0
          DO 5234, I=1,NMAT
             K=K+2
             MAXVEC(I,1)=X(K-1)
             MAXVEC(I,2)=X(K)
5234     CONTINUE
          amaxamp=L-1.0
          SECOND=1
          TSKP=0
          GOTO 2201
          ENDIF
          IF(SECOND.EQ.0)THEN
```

```
          SECOND=2
          GOTO 2201
          ENDIF
4004      DO 127 J=1,N
127          Y(J)=X(J)
          SKP=MCLOCK()-TIME1-starttime
          TMYA=I-1.0
          ral=amaxamp
          EV=NXTEV
          EIGENSYS(5)=TIME1
          K=0.0
          DO 1234, I=1,NMAT
             K=K+2
             EIGVEC(I,1)=Y(K-1)
             EIGVEC(I,2)=Y(K)
1234      CONTINUE
C
C***********    TASK THREE AND FOUR ***    TASK THREE AND FOUR **********
C
C TAKES THE EIGENVECTOR AND MINIMUM EIGENVALUE AND SOLVES
C THE FUCTION FOR PEAKS VIA THE TWO EIGENVECTOR ESTIMATOR
C
          DOWN=1.0
          AMAXAMP=1000.0
          PI=3.141592654
          FINISH=1.570796327
          INC=.001745329252
          NUM=INT(1800.0/IZE)
          START=NUM*(IZE-1)*INC-FINISH
1010      L=NINT(START/INC)+900
          PEAK(L)=1000000000000000.0
          DO 910 T=START,FINISH,INC
             L1=L
             T1=T-INC
             L=L+1
             L2=L+1
             PLOT(L)=0.0
             PLOT(L2)=0.0
             PEAK(L)=0.0
             PEAK(L2)=0.0
             NSEP=SEP
             DO 20 I=1,NMAT
                AG=((((I-((NMAT+1.0)/2.0))*PI*2*NSEP)/WVLTH)*SIN(T))
                AATVAL(I,1)=COS(AG)
                AATVAL(I,2)=SIN(AG)
20           CONTINUE
             DO 30 I=1,NMAT
                PLOT(L)=PLOT(L)+(MAXVEC(I,1)*AATVAL(I,1)
     >                    +MAXVEC(I,2)*AATVAL(I,2))
                PLOT(L2)=PLOT(L2)+(MAXVEC(I,1)*AATVAL(I,2)
     >                    -MAXVEC(I,2)*AATVAL(I,1))
                PEAK(L)=PEAK(L)+(EIGVEC(I,1)*AATVAL(I,1)
     >                    +EIGVEC(I,2)*AATVAL(I,2))
                PEAK(L2)=PEAK(L2)+(EIGVEC(I,1)*AATVAL(I,2)
     >                    -EIGVEC(I,2)*AATVAL(I,1))
30           CONTINUE
C
             PEAK(L)=((PLOT(L)*plot(1)+plot(12)*PLOT(L2))**2)/
```

```
     >              (PEAK(L)*peak(I)+peak(I2)*PEAK(L2))
            IF(PEAK(L).GT.AMAXAMP)THEN
              IF(PEAK(L).GT.PEAK(L1))THEN
                 DOWN=-1.0
                 GOTO 910
               else
                 IF (DOWN.GT.0.0)GOTO 910
                    DO 6910 PK=NMSIG,1,-1
                       IF (PEAK(L1).LT.ODOA(PK))THEN
                          GOTO 6910
                       ELSE
                          ODOA(PK+1)=ODOA(PK)
                          OADOA(PK+1)=OADOA(PK)
                          ODOA(PK)=PEAK(L1)
                          OADOA(PK)=T1
                       ENDIF
6910                CONTINUE
                    DOWN=1.0
              ENDIF
            endif
910         CONTINUE
C
            DO 388 J=1,DIM
C
                  OUTTYPE=2**(J-1)
                  CALL CRECV(OUTTYPE,OUTMSG,OUTSIZE)
C
              DO 8910 PK=1,NMSIG
                 DO 8910 K=NMSIG,1,-1
                    IF (DOA(PK).LT.ODOA(K)) GOTO 8910
                       ODOA(K+1)=ODOA(K)
                       OADOA(K+1)=OADOA(K)
                       ODOA(K)=DOA(PK)
                       OADOA(K)=ADOA(PK)
8910         CONTINUE
                  DO 388 I=START,START+((2**(J-1))*NUM)-1
                     PEAK(I)=PLOT(I)
388               CONTINUE
                  DO 3899 I=1,10
                   OADOA(I)=OADOA(I)/.01745329252
3899              CONTINUE
                  OUTTYPE=35
C
C******** STOP THE TIMER ALL WORK DONE ************************
C
                OSTART=MCLOCK()-SKP-time1-starttime
C
C
C*********END OF NODE PARALLEL PROCESSING **** ONLY OUTPUT LEFT*********
C
            CALL CSEND(30,EIGENSYS, EIGSIZE, HOST, HOSTPID)
C
            CALL CSEND(OUTTYPE,OOUTMSG,OUTSIZE,HOST,HOSTPID)
C
            CALL FLUSHMSG(-1,-1,0)
C
            STOP
C*******************************************************************************
```

```
C  THIS SECTION IS FOR ALL WORKING NODES.
C  THIS SECTION TAKES THE SHIFT,S, AND SUBTRACTCS IT FROM THE
C  DIAGONAL OF THE MATRIX THE STEERING IS DONE WHEN S=EV
C   INDICATES THIS ACTION ,ELSESHIFT IS ZERO, EXIT.
C
99         INITSIZE=4*(NCPN*320)
           PARAMSIZE=PSZ
           CALL CRECV(PARAMTYPE,PARAMSG(1),PARAMSIZE)
           CALL CRECV(SECDTYPE, SECDMSG(1), INITSIZE)
           CALL CRECV(THRDTYPE, THRDMSG(1), INITSIZE)
           CALL CRECV(STARTTYPE, STARTMSG, 4)
           S    =0.0
           TSKP=0.0
           SKP  =0.0
           D    =0.0
C
           MYB=(MYNOD)*NCPN
           MYA=MYB+1-NCPN
C
C*******     TASK ONE   ***    TASK ONE ***    TASK ONE ********************
C
C  THIS ROUTINE CONVERTS THE SAMPLE MATRIX INTO A NXN
C  MATRIX THAT ESTIMATES THE EXPECTED VALUE
C  OF THE SAMPLES ALLOWING A TEMORAL AVERAGING PROCESS TO
C   IMPROVE THE SIGNAL TO NOISE RATIO
C
          DO 23410 J=1,NMSP,4
             J2=J+1
             J3=J2+1
             J4=J3+1
             DO 23410 K=1,N,2
                DO 23410 L=K,N
                   VAL1=Z(K,J)*Z(L,J)+Z(K,J3)*Z(L,J3)
                   VAL2=Z(K,J2)*Z(L,J2)+Z(K,J4)*Z(L,J4)
                   XD(K,L)=XD(K,L)+VAL1+VAL2
23410        CONTINUE
C
C WRAPPING UP TO RECEIVE AND ADD TO MATRICES COMPUTED HERE
           K=0
           FINISH=N*320
           INC=2*320+2
           START=-INC+1
           DO 29342 J=1,NMAT
              START=START+INC
              DO 29342 I=START,FINISH,320
                 K=K+1
                 XDS(K)=XDS(I)
29342       CONTINUE
C
            DO 23588 I=1,DIM
              IF ((MOD(MYNOD,2**I)).NE.0)GOTO 2342
              CALL CRECV(COVTYPE,COVMSG(1),COVSIZE)
              DO 23588 K=1,SCOVSIZE
                    XDS(K)=XDS(K)+AS(K)
23588       CONTINUE
2342        DEST=MYNOD-(2**(I-1))
            CALL CSEND(COVTYPE,COVUPMSG(1), COVSIZE, DEST, PID)
C
```

```
            K=DIM
            DO 23431 I=1,DIM
               K=K-1.0
               IF (MOD(MYNOD,2**K).EQ.0)GOTO 23441
23431       CONTINUE
23441       FCOVSIZE=NCPN*(320*4)*2**K
            CALL CRECV(FCOVTYPE,COVMSG(1),FCOVSIZE)
            DO 2333 I=1,K
               FCOVSIZE=NCPN*(320*4)*2**(K-I)
               DEST=MYNOD+2**(K-I)
               L=(320*NCPN*(2**(K-I)))+1
               CALL CSEND(FCOVTYPE,COVMSG(L),FCOVSIZE,DEST,0)
2333        CONTINUE
C
          DO 343 K=1,N-1,2
             K2=K+1
             DO 343 L=1,NCPN-1,2
                L2=L+1
                A(K2,L)=-A(K,L2)
                A(K2,L2)=A(K,L)
343       CONTINUE
C
C**********    TASK 2    ***    TASK 2    ***    TASK 2    ******************
C
C  ROUTINE EIGPOWER TAKES THE A MATRIX, WHICH MAY BE SHIFTED
C  AND OUTPUTS THE ESTIMATED
C  EIGENVECTOR, AND EIGENVALUE.
C
            TIMES=TIMES+1
201         TIMES=TIMES-1
            IF (TIMES.LT.0.0)GOTO 1000
               IF((S.NE.0.0).AND.(D.EQ.0.0))THEN
                  DO 10 J=1,N
                       DO 10 I=1,NCPN
                          A(J,I)=A(J,I)-S*C(J,I)
10                CONTINUE
                  D=1
               ENDIF
               SKP=0.0
               IF(GEN.EQ.0.0)THEN
                 K=0
                 DO 51876 I=MYA,MYB
                    Y(I)=0.0
                    K=K+1
                    DO 51876 J=1, N
                       Y(I)=Y(I),+X(J)*A(J,K)
51876               CONTINUE
               ELSE
                  K=0
                  DO 5456 I=1,N
                     Y2(I)=0.0
5456              CONTINUE
                  DO 76 I=MYA,MYB
                        K=K+1
                        DO 76 J=1, N
                         Y2(I)=Y2(I)+X(J)*A(J,K)
76                CONTINUE
C
```

```
            DO 2532 J=1,DIM
            IF ((MOD(MYNOD,2**J)).NE.0)GOTO 2533
                YTYPE=MYNOD+2**(J-1)
                CALL CRECV(YTYPE,MY,YSIZE)
                DO 2532 I=1,N
                   Y2(I)=Y2(I)+MY(I)
2532        CONTINUE
2533        DEST=MYNOD-2**(J-1)
            CALL CSEND(MYNOD,Y2,YSIZE,DEST, PID)
            CALL CRECV(99,Y2,YSIZE)
C
            K=0
            DO 86 I=MYA,MYB
                  Y(I)=0.0
                  K=K+1
                  DO 86 J=1,N
                     Y(I)=Y(I)+Y2(J)*B(J,K)
86          CONTINUE
            ENDIF
C
C  LOOK FOR LARGEST VALUE TO NORMALIZE VECTOR.  THIS WILL
C  BE THE ESTIMATE FOR THE EIGENVALUE.
C
          EV=Y(MYA)
          DO 1130 J=MYA+1,MYB
                IF(ABS(EV).LT.ABS(Y(J)))EV=Y(J)
1130        CONTINUE
41        DO 588 I=1,DIM
                IF ((MOD(MYNOD,2**I)).EQ.0) GOTO 587
                   DEST=MYNOD-(2**(I-1))
                   TMYA=MYA
                   EIGTYPE=MYNOD
C
42        CALL CSEND(EIGTYPE,EIGENSYS, EIGSIZE, DEST, PID)
C
          CALL CRECV(ITERTYPE, ITERMSG, ITERSIZE)
          IF (SECOND.EQ.1)THEN
              TIMES=200.0
              K=0.0
              DO 3234, J=1,NMAT
                 K=K+2
                 MAXVEC(J,1)=X(K-1)
                 MAXVEC(J,2)=X(K)
3234          CONTINUE
              DO 1641 J=1,N
                 X(J)=1.0
1641          CONTINUE
          ENDIF
          IF (SECOND.EQ.2)GOTO 1000
          GOTO 201
C
587       DO 511 J=MYA,MYA+(NCPN*2**(I-1))-1
             X(J)=Y(J)
511       CONTINUE
          TSKP=SKP
          NXTEV=EV
          EIGTYPE=MYNOD+2**(I-1)
          CALL CRECV(EIGTYPE,EIGENSYS,EIGSIZE)
```

```
C
          TSKP=TSKP+SKP
C
          IF (ABS(NXTEV).GT.ABS(EV)) EV=NXTEV
C
          DO 581 K=MYA,TMYA-1
              Y(K)=X(K)
581       CONTINUE
          SKP=TSKP
588       CONTINUE
          STOP
C
C
C***********   TASK THREE AND FOUR ***   TASK THREE AND FOUR **********
C
C TAKES THE MAX EIGENVECTOR AND MINIMUM EIGENVECTOR AND SOLVES
C THE FUCTION FOR PEAKS VIA TJR TWO-EIGENVAVECTOR ESTEMATOR
C
1000      K=0
          DO 91234, I=1,NMAT
             K=K+2
             EIGVEC(I,1)=X(K-1)
             EIGVEC(I,2)=X(K)
91234     CONTINUE
          DOWN=1.0
          MAXAMP=1000.0
          PI=3.141592654
          FINISH=1.570796327
          INC=.001745329252
          NUM=INT(1800.0/IZE)
          START=(MYNOD-1.0)*NUM*INC-FINISH
          FINISH=START+(NUM+1)*INC
91010     L=NINT(START/INC)+900
          PLOT(L)=1000000000000.0
          DO 9 T=1813,1834
             OUTMSG(T)=0.0
9         CONTINUE
          DO 9910 T=START,FINISH,INC
             L1=L
             T1=T-INC
             L=L+1
             L2=L+1
             PLOT(L)=0.0
             PLOT(L2)=0.0
             PEAK(L)=0.0
             PEAK(L2)=0.0
             NSEP=SEP
             DO 920 I=1,NMAT
                AG=((((I-((NMAT+1.0)/2.0))*PI*2*NSEP)/WVLTH)*SIN(T))
                AATVAL(I,1)=COS(AG)
                AATVAL(I,2)=SIN(AG)
920          CONTINUE
             DO 930 I=1,NMAT
                PLOT(L)=PLOT(L)+(MAXVEC(I,1)*AATVAL(I,1)
     >                    +MAXVEC(I,2)*AATVAL(I,2))
                PLOT(L2)=PLOT(L2)+(MAXVEC(I,1)*AATVAL(I,2)
     >                    -MAXVEC(I,2)*AATVAL(I,1))
                PEAK(L)=PEAK(L)+(EIGVEC(I,1)*AATVAL(I,1)
```

```
     >                    +EIGVEC(I,2)*AATVAL(I,2))
           PEAK(L2)=PEAK(L2)+(EIGVEC(I,1)*AATVAL(I,2)
     >                    -EIGVEC(I,2)*AATVAL(I,1))
930         CONTINUE
C
           PLOT(L)=((PLOT(L)*plot(I)+plot(I2)*PLOT(L2))**2)/
     >             (PEAK(L)*peak(I)+peak(I2)*PEAK(L2))
           IF(PLOT(L).GT.MAXAMP)THEN
           IF(PLOT(L).GT.PLOT(L1))THEN
             DOWN=-1.0
             GOTO 9910
           ELSE
             IF (DOWN.GT.0.0) GOTO 9910
               DO 29910 PK=NMSIG,1,-1
                  IF (PLOT(L1).LT.DOA(PK))THEN
                     GOTO 29910
               ELSE
                  DOA(PK+1)=DOA(PK)
                  ADOA(PK+1)=ADOA(PK)
                  DOA(PK)=PLOT(L1)
                  ADOA(PK)=T1
             ENDIF
29910      CONTINUE
             DOWN=1.0
           ENDIF
           ENDIF
9910       CONTINUE
C
           SSTART=NINT(START/INC)+901
741        DO 7588 I=1,DIM
                 IF ((MOD(MYNOD,2**I)).EQ.0) GOTO 7587
                    DEST=MYNOD-(2**(I-1))
                    START=SSTART
                    OUTTYPE=MYNOD
742        CALL CSEND(OUTTYPE,OUTMSG,OUTSIZE, DEST, PID)
C
           STOP
C
7587       DO 7511 J=SSTART,SSTART+(NUM*2**(I-1))
              PEAK(J)=PLOT(J)
7511       CONTINUE
           DO 2751 J=1813,1834
              PEAK(J)=PLOT(J)
2751       CONTINUE
           OUTTYPE=MYNOD+(2**(I-1))
           CALL CRECV(OUTTYPE,OUTMSG,OUTSIZE)
C
           DO 18910 PK=1,NMSIG
              DO 18910 K=NMSIG,1,-1
                 IF (DOA(PK).LT.ODOA(K))GOTO 18910
                    ODOA(K+1)=ODOA(K)
                    OADOA(K+1)=OADOA(K)
                    ODOA(K)=DOA(PK)
                    OADOA(K)=ADOA(PK)
18910       CONTINUE
C
              DO 1588 J=START,START+((2**(I-1))*NUM)-1
                 PEAK(J)=PLOT(J)
```

```
1588            CONTINUE
                DO 5678 J=SSTART,START+((2**(I-1))*NUM)
                   PLOT(J)=PEAK(J)
5678            CONTINUE
                DO 4352 J=1813,1834
                   PLOT(J)=PEAK(J)
4352            CONTINUE
7588            CONTINUE
C
C********END OF NODE PARALLEL PROCESSING **** ONLY OUTPUT LEFT*********
C
         STOP
C
C*********************  STOP STOP   STOP  ***********************
C
C  THIS IS A NODE IDLE CONDITION TO RECEIVE MSG AND NOT STOP CUBE
C
100       CALL CRECV(ITERTYPE,ITERMSG,ITERSIZE)
          IF(SECOND.EQ.2)THEN
          PRINT *,'STOPPED NODE=',MYNOD
          STOP
          ENDIF
          GOTO 100
C
            END
```

VITA

Antone L. Kusmanoff

Candidate for the Degree of
Doctor of Philosophy

Thesis: REAL-TIME BEARING ESTIMATON IN A MULTI-SOURCE ENVIROMENT USING MULTI-PROCESSOR, MULTI-ALGORITHMIC ACCELERATION

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Alton, Illinois, November 22, 1943, the son of Boris M. and Eveyln J. Kusmanoff

Education: Graduated from East Alton Wood River Community High School, Wood River, Illinois, in June 1961; received Bachelor of Arts in Math from Southern Illinois University at Carbondale in May, 1967; received Bachelor of Science Degree in Electrical Engineering from the University of Missouri at Columbia in Dec, 1972; received Master of Science Degree in Electrical Engineering from the University of Missouri at Columbia in May, 1973; received Master of Science Degree in Electrical Engineering from Georgia Institute of Technology at Atlanta in May,1982; Completed requirements for the Doctor of Philosophy degree at Oklahoma State University at Stillwater in May, 1989.

Professional Experience: Commissioned into the United States Air Force as a Second Lieutenant in June 1967, retired as a Lieutenant Colonel in July 1987. Performed graduate and undergraduate instructional duties at Chapman College in 1978-1980, at Georgia Institute of Technology in 1980-1982, at The Air Force Institute of Technology 1982-1984, at University of Southern Mississippi in 1984-87, and at Oklahoma State University between 1987-1989. Senior Research Engineer at Southwest Research Institute in 1989. Member of Tau Beta Pi and Eta Kappa Nu honorary societies.